mathematics

MDPI

Article

Energy-Efficient Hybrid Flowshop Scheduling with Consistent Sublots Using an Improved Cooperative Coevolutionary Algorithm

Chengshuai Li [1], Biao Zhang [1,*], Yuyan Han [1,*], Yuting Wang [1], Junqing Li [2] and Kaizhou Gao [3]

[1] School of Computer Science, Liaocheng University, Liaocheng 252059, China
[2] School of Computer Science, Shandong Normal University, Jinan 252000, China
[3] Macau Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau 999078, China
* Correspondence: zhangbiao@lcu-cs.com (B.Z.); hanyuyan@lcu-cs.com (Y.H.); Tel.: +86-13863565273 (B.Z.); +86-18864974734 (Y.H.)

Abstract: Energy conservation, emission reduction, and green and low carbon are of great significance to sustainable development, and are also the theme of the transformation and upgrading of the manufacturing industry. This paper concentrates on studying the energy-efficient hybrid flowshop scheduling problem with consistent sublots (HFSP_ECS) with the objective of minimizing the energy consumption. To solve the problem, the HFSP_ECS is decomposed by the idea of "divide-and-conquer", resulting in three coupled subproblems, i.e., lot sequence, machine assignment, and lot split, which can be solved by using a cooperative methodology. Thus, an improved cooperative coevolutionary algorithm (vCCEA) is proposed by integrating the variable neighborhood descent (VND) strategy. In the vCCEA, considering the problem-specific characteristics, a two-layer encoding strategy is designed to represent the essential information, and a novel collaborative model is proposed to realize the interaction between subproblems. In addition, special neighborhood structures are designed for different subproblems, and two kinds of enhanced neighborhood structures are proposed to search for potential promising solutions. A collaborative population restart mechanism is established to ensure the population diversity. The computational results show that vCCEA can coordinate and solve each subproblem of HFSP_ECS effectively, and outperform the mathematical programming and the other state-of-the-art algorithms.

Keywords: hybrid flowshop scheduling; energy efficiency; consistent sublots; collaborative coevolutionary algorithm; variable neighborhood descent

MSC: 90B30

Citation: Li, C.; Zhang, B.; Han, Y.; Wang, Y.; Li, J.; Gao, K. Energy-Efficient Hybrid Flowshop Scheduling with Consistent Sublots Using an Improved Cooperative Coevolutionary Algorithm. *Mathematics* **2023**, *11*, 77. https://doi.org/10.3390/math11010077

Academic Editor: Ioannis G. Tsoulos

Received: 18 November 2022
Revised: 16 December 2022
Accepted: 22 December 2022
Published: 25 December 2022

1. Introduction

With the changing climate and environment, green development, energy saving, and emission reduction become the themes of transformation and upgrading of the manufacturing industry. Advanced production scheduling technology can effectively improve production efficiency and reduce energy consumption in the manufacturing industry, enhancing the core competitiveness of enterprises. As a branch of scheduling problems, the hybrid flowshop scheduling problem (HFSP) [1] has a very strong industrial application background, such as microelectronics, furniture, textile, petrochemical, and pharmaceutical fields [2–5]. In HFSP, a group of jobs need to go through a series of processing stages in succession and each stage has multiple identical machines. The goal of the HFSP is to determine the job sequence and machine assignment of these jobs at each stage with considering production constraints. The problem is a very complex combinatorial optimization problem [6], and even on a very small scale, it proves to be NP-hard [1]. In the

most research on the HFSP [7], each job is treated as a whole, and the job cannot proceed to the next stage before the completion of the processing at a given stage [8]. In the actual production scenario [9,10], a job, called a lot in the following, usually consists of a number of identical items. When the lot is large, items already processed completely on a machine need to wait a long time in the output buffer of this machine, whereas the downstream machine may be idle. This scenario will have a negative impact on the production efficiency and lead to unnecessary energy consumption. Therefore, it is very important to develop a scheduling methodology suitable for this scenario to enhance the energy efficiency and core competitiveness of such factories.

In this paper, we introduce the technique of lot streaming into the HFSP, resulting in a novel problem, i.e., lot streaming HFSP. The lot streaming, first introduced by Reiter [11] in the context of job shop scheduling, is preferable for implementing the time-based strategy and widely adopted by top-notch companies to improve their customer service. Lot streaming is the process of splitting a large lot into several sublots and scheduling those sublots in an overlapping fashion to accelerate progress [12]. That is, the lot streaming is used to divide a lot with a large number of items into several sublots with a small number of items. Each sublot can be transported to the downstream stage for processing immediately after its completion at the upstream stage and does not have to wait for the completion of the entire lot. This method can effectively reduce the production cycle and improve production efficiency so that products can be delivered to customers faster, and more orders can be accepted within a limited time. Moreover, this method can effectively increase machine utilization, reduce machine idle time, and thus reduce energy consumption.

According to the lot streaming studies, the lot division methods [13] are equal sublots, consistent sublots, and variable sublots. With equal sublots, a lot can be divided into several sublots with equal size, i.e., each sublot contains the same number of items, and the number and size of sublots remain unchanged throughout the processing process. Consistent sublots mean that a lot is divided into several sublots that may have different sizes, and the number and size of sublots remain unchanged throughout the processing process. Equal sublots can also be understood as a specific case of consistent sublots. Unlike consistent sublots, in variable sublots [14], the number and size of sublots may change throughout the processing process. In real production, variable sublots are rarely used because their diverse nature seriously increases the difficulty of production management. Moreover, its comprehensive cost performance is not high for most enterprises. In contrast, consistent sublots are often used in most enterprises' actual production.

In sum, the energy-efficient HFSP with consistent sublots (HFSP_ECS) is the focus of our study. To solve the problem, three coupled subproblems must be addressed, i.e., lot sequence, machine assignment, and lot split. Thus, the HFSP_ECS is much more complex than the classical HFSP, and obviously NP-hard. With its NP-hard property, the metaheuristics are suggested to solve the problem. In addition, when using the metaheuristics, in order to obtain a globally optimal solution, the three subproblems must be coevolved and addressed simultaneously [15,16]. It is therefore natural to employ the cooperative coevolutionary algorithm (CCEA) [17]. Its design is inspired by the natural phenomenon that the coexisting species promote each other and coevolve. The algorithm adopts the strategy of "divide and conquer", which decomposes an optimization problem into several subproblems. In addition, the whole problem is optimized by a reciprocal evolutionary mechanism driven by cooperative or competitive interactions between subproblems [18]. The local search strategy also plays an important role in CCEA. This paper develops an improved cooperative coevolutionary algorithm (vCCEA) by integrating the variable neighborhood descent (VND) strategy [19]. The vCCEA can solve the whole problem by evolving the subproblems simultaneously and interacting between the subproblems. In addition, considering the problem-specific characteristics, a two-layer encoding strategy is designed to represent the solution information and a novel collaborative model is proposed to realize the interaction between subproblems. Special neighborhood structures are designed for different subproblems and two kinds of enhanced local disturbance strategies are pro-

posed to search for potential promising solutions. This algorithm mainly contains four processes, i.e., initialization process, cooperative coevolutionary process, VND processes, and population restart processes. First, an archive that holds several complete solutions is initialized and two populations based on these solutions are built in the initialization process. Then, the two populations and archive coevolve through the collaborative model in the coevolutionary process. While in the cooperative coevolutionary process, the VND process is used to generate a new solution. With the evolution proceeding, the population restart process can be triggered to ensure the population diversity.

The main contributions of this study are as follows. (1) An energy-efficient hybrid flowshop scheduling problem with consistent sublots (HFSP_ECS) is studied and a mathematical model is developed for it. (2) An improved cooperative coevolutionary algorithm based on the idea of "divide-and-conquer" is proposed by integrating the VND strategy. (3) A novel collaborative model suitable for the specific characteristics of HFSP_ECS is designed to realize the interaction between the populations and the archive. (4) Two kinds of enhanced local neighborhood structures are proposed to search for potential promising solutions.

The remaining of the paper is organized as follows. A brief literature review is provided in Section 2. Section 3 describes HFSP_ECS in detail and a linear integer programming model (MILP) is established for a better representation of this problem. Section 4 introduces the algorithm process of vCCEA and improvement strategies in detail. In Section 5, the experimental study design is presented and the results are analyzed. Finally, some conclusions are given and future research prospects are outlined in Section 6.

2. Literature Review

Although HFSP has been studied for several decades, little research has been carried out on energy-efficient HFSP with lot streaming. Most of the existing studies have been conducted with the objective of minimizing the production cycle to optimize HFSP with lot streaming, and little attention has been paid to the energy consumption in the production process. The following is a first review of the HFSP with lot streaming in detail, and then the existing research results on green scheduling are analyzed. Finally, the characteristics of the research problem in this paper are summarized.

With the development of a multi-species small-scale production model in recent years, more and more scholars are focusing on the HFSP with lot streaming. Depending on the number of lots, the lot streaming HFSP can be divided into two main categories, i.e., single-lot HFSP and multiple-lot HFSP. The single-lot HFSP means that only one lot needs to be processed, and how to divide lots and how to sort sublots are two major problems, i.e., sublot size and sublot sequence. Zhang et al. [12] studied a special two-stage HFSP with single-lot that the first stage has multiple identical machines and the second stage only has one machine. They first formulated the problem as an MILP considering the equal sublots, and proposed a heuristic to reach an effective solution. For the same problem, Liu [20] used linear programming and rotation methods to solve the sublot sequence and sublot size, respectively. Moreover, an effective heuristic rule is proposed for the general HFSP with equal sublots. Cheng et al. [21] studied a two-stage HFSP that the first stage only has one machine and the second stage has two parallel machines. Assuming that the number of sublots are known, the closed-form expressions are used to obtain the best sublot sizes. Then, according to the best sublot sizes, the upper bound of the sublot quantities is defined, and an algorithm combining closed-form expressions is used to obtain the global optimal solution. In addition, a heuristic is proposed for the case where the number of sublots is unknown.

Compared with the single-lot HFSP, more research focuses on multiple-lot HFSP. Potts and Baker [22] first showed how to use equal sublots in the one-job model and analyzed equal-sized sublots as a heuristic procedure. After that, they cited some difficulties in multiple-lot scheduling. Kalir and Sarin [23] studied a multiple-lot HFSP with small equal sublots, and proposed a heuristic called bottleneck minimal idleness with the ob-

jective of minimizing the maximum completion time. Naderi and Yazdani [24] studied a multiple-lot HFSP with setup time constraints. Assuming that the number of sublots were known, an MILP was established and an imperialist competitive algorithm was proposed. Zhang et al. [25] studied the HFSP with equal sublots, and a discrete fruit fly optimization algorithm was developed for solving this problem, where two main search procedures were designed to balance the exploration and exploitation abilities of the algorithm. For the same problem, Zhang et al. [26] proposed an effective migrating birds optimization algorithm with the objective of minimizing the total flow time, and a heuristic rule was introduced to address the case that the sublots from different lots have the changes to reach the downstream stage at the same time.

The multiple-lot HFSP studied above were all with equal sublots, and this means that sublots from the same lot have the equal size. When the sublots from the same lot are not equal in size, the multiple-lot HFSP is called HFSP with consistent sublots. For example, Ming Cheng and Sarin [13] studied a two-stage HFSP where the first stage only had one machine and the second stage had two identical machines. They used some conclusions from the single-lot scheduling problem, and proposed a mathematical programming-based heuristic method for this problem. Zhang et al. [27] studied a special two-stage HFSP where the first stage had multiple identical machines and the second stage only had a single machine. Additionally, two heuristic strategies were proposed to solve two subproblems, i.e., lot sequence, and lot split. Nejati et al. [28] studied a multiple-lot k-stage HFSP with a specific production scenario. They improved the genetic algorithm and simulated an annealing algorithm for this particular problem, and the effectiveness of the improved strategy was verified. Lalitha et al. [29] studied a special k-stage HFSP where the front k-1 stages only had one machine per stage and the last stage had multiple machines. An MILP was developed and some small-scale problems were solved by the optimizer. A two-stage heuristic algorithm was proposed to solve medium–large scale problems, hierarchically. Zhang et al. [30] studied an HFSP with consistent sublots and considered the setup and transportation operations. A collaboration operator was proposed and a collaborative variable neighborhood descent algorithm was developed based on this operator.

Green development, energy saving, and emission reduction are of great significance to sustainable development. Qin et al. [31] studied an HFSP with an energy-saving criterion, and considered blocking constraints. A mathematical model for HFSP with blocking constraints and energy-efficient criterion was developed and an improved iterative greedy algorithm based on the swap operator was proposed. Duan et al. [32] studied a heterogeneous multi-stage HFSP with energy-efficient for large metal component manufacturing, and an improved NSGA-II combined with the moth-flame optimization algorithm (NSGA–II–MFO) with the objective of minimizing the maximum completion time and carbon emission was proposed. Dong et al. [33] studied a distributed two-stage re-entrant green HFSP, a two-level mathematical model and an improved hybrid slap swarm and NSGA-III algorithm with the objectives of minimum completion time, total carbon emission and total energy consumption was proposed. Geng et al. [34] studied an energy-efficient re-entrant HFSP with considering customer order constraints under Time-of-Use (TOU) electricity price, and a memetic algorithm with an energy saving strategy was proposed to solve this problem.

In summary, both the lot streaming HFSP and the green HFSP have had a certain number of research results. Compared with these studies, the characteristics of our study can be summarized as follows. A k-stage energy-efficient HFSP with consistent sublots is studied in this paper, and the number of machines per stage is not limited. While Ming Cheng and Sarin [13] and Wei Zhang et al. [27] studied the special two-stage HFSP, and Lalitha et al. [29] studied a special k-stage HFSP that the first k-1 stages only have one machine at each stage and the last stage has multiple machines. Compared with these studies, the HFSP_ECS studied in this paper has a wider scope of application. In study of Ming Cheng and Sarin [13] and Naderi and Yazdani [24], the sublots from different lots can be mixed and cross-processed, but they are prohibited in our research. This is because in

real production, the machine needs to be adjusted accordingly before processing different products. Assuming that sublots are allowed to be mixed, the machine will be in a state of frequent adjustment, which has a serious impact on productivity and increases unnecessary energy consumption. Additionally, in the above studies on lot streaming, the research focuses on minimizing the completion time without considering the energy consumption in the process. However, in the actual production, energy consumption is a non-negligible factor. In our study, all machines were turned on and off uniformly, and there was a positive correlation between energy consumption and minimized completion time.

3. Problem Description

The HFSP_ECS addressed can be described as follows. A set of lots J is to be consecutively processed in a series of K stages. Each stage k has $M_i \geq 1$ identical parallel machines and at least one stage has the number of machines greater than one, i.e., $M_i > 1$. Each lot to be processed is made up of a group of identical items. The consistent sublots is employed to split a lot to several sublots with assuming that the maximum sublot quantities are limited. Each sublot contains a certain number of items, and the number of the items contained in a sublot is defined as the sublot size. The number and size of the sublots do not change during the K processing stages. At the same stage, different sublots from the same lot are processed continuously on the same machine. Similarly, the items from the same sublot need to be processed continuously. The sublots can proceed to the next stage immediately after its completion of the previous stage. The processing time of a sublot is the product of the sublot size and the item processing time. The processing energy consumption of the sublot is the product of the unit energy consumption and processing time. The idle energy consumption of a machine is the product of unit idle energy and idle time, the idle time, and the idle energy consumption per unit. The scheduling task of the HFSP_ECS is to solve the three subproblems' lot sequence, machine assignment, and lot split, and its objective is to minimize the energy consumption. The assumptions are summarized as follows:

- All machines are available at time 0, and all machines turn off uniformly at the end of the process.
- Assume an infinite buffer between stage and allow the machine to be idle.
- Each lot must be processed through all stages, and only one machine can be selected at the same stage, and interrupt and preemption are not allowed during processing.
- One machine can at most process only one item at the same time, and the items from the same sublot need to be processed continuously.
- Each lot is divided into several sublots and the sublot quantities are limited by a maximum value.
- The sublots of each lot can be processed at the next stage immediately after the completion of the previous stage.
- The first sublot can be started as soon as it arrives at this stage. After the remaining sublots reach the stage, it also needs to wait for the previous sublots to complete processing before it can be processed.
- Sublots from different lots are not allowed to be mixed during processing; if two lots are processed on the same machine, the later lot will not be processed until all the sublots of the previous lot have been processed.
- Machine setup and transport operations are included in the machining process.

With the above description and assumptions, to better describe and solve this problem, an MILP [30] is established, the notations and constraints are described as follows:

Objective:

$$Minimize(E_{\max}) \tag{1}$$

Constraints:

$$C_{\max} \geq C_{K,j,L} \quad \forall j \in \{1, 2, \ldots, J\} \tag{2}$$

$$\sum_{i=1}^{M_k} D_{k,j,i} = 1 \quad \forall k \in \{1,2,\ldots,M_k\}, \forall j \in \{1,2,\ldots,J\} \tag{3}$$

$$\sum_{e=1}^{L} N_{j,e} = T_j \quad \forall j \in \{1,2,\ldots,J\}, \forall e \in \{1,2,\ldots,L\} \tag{4}$$

$$N_{j,e} \geq 0 \quad \forall j \in \{1,2,\ldots,J\}, \forall e \in \{1,2,\ldots,L\} \tag{5}$$

$$N_{j,e} + (1 - W_{j,e}) \times G \geq 1 \quad \forall j \in \{1,2,\ldots,J\}, \forall e \in \{1,2,\ldots,L\} \tag{6}$$

$$N_{j,e} - W_{j,e} \times G \leq 0 \quad \forall j \in \{1,2,\ldots,J\}, \forall e \in \{1,2,\ldots,L\} \tag{7}$$

$$W_{j,e} \geq W_{j,e+1} \quad \forall j \in \{1,2,\ldots,J\}, \forall e \in \{1,2,\ldots,L-1\} \tag{8}$$

$$S_{1,j,1} \geq 0 \quad \forall j \in \{1,2,\ldots,J\} \tag{9}$$

$$C_{k,j,e} = S_{k,j,e} + P_{k,j} \times N_{j,e} \quad \forall k \in \{1,2,\ldots,K\}, \forall j \in \{1,2,\ldots,J\}, \forall e \in \{1,2,\ldots,L\} \tag{10}$$

$$S_{k+1,j,e} - C_{k,j,e} \geq 0 \quad \forall k \in \{1,2,\ldots,K-1\}, \forall j \in \{1,2,\ldots,J\}, \forall e \in \{1,2,\ldots,L\} \tag{11}$$

$$S_{k,j,e+1} - C_{k,j,e} \geq 0 \quad \forall k \in \{1,2,\ldots,K\}, \forall j \in \{1,2,\ldots,J\}, \forall e \in \{1,2,\ldots,L-1\} \tag{12}$$

$$Y_{k,j,j1,i} + Y_{k,j1,j,i} \leq D_{k,j,i} \quad \forall k \in \{1,2,\ldots,K\}, \forall j! = j1, j \in \{1,2,\ldots,J\}, j1 \in \{1,2,\ldots,J\}, \forall e \in \{1,2,\ldots,L-1\}, i \in \{1,2,\ldots,M_k\} \tag{13}$$

$$Y_{k,j,j1,i} + Y_{k,j1,j,i} \leq D_{k,j1,i} \quad \forall k \in \{1,2,\ldots,K\}, \forall j! = j1, j \in \{1,2,\ldots,J\}, j1 \in \{1,2,\ldots,J\}, i \in \{1,2,\ldots,M_k\} \tag{14}$$

$$Y_{k,j,j1,i} + Y_{k,j1,j,i} \geq D_{k,j,i} + D_{k,j1,i} - 1 \quad \forall k \in \{1,2,\ldots,K\}, \forall j! = j1, j \in \{1,2,\ldots,J\}, j1 \in \{1,2,\ldots,J\}, i \in \{1,2,\ldots,M_k\} \tag{15}$$

$$S_{k,j,1} - C_{k,j1,L} + G \times (3 - Y_{k,j1,j,i} - D_{k,j,i} - D_{k,j1,i}) \geq 0 \quad \forall k \in \{1,2,\ldots,K\}, \forall j! = j1, j \in \{1,2,\ldots,J\}, j1 \in \{1,2,\ldots,J\}, i \in \{1,2,\ldots,M_k\} \tag{16}$$

$$E_{process} = \sum_{k=1}^{K} \sum_{j=1}^{J} \sum_{e=1}^{L} N_{j,e} \times EP_{k,j} \tag{17}$$

$$E_{idle} = \sum_{k=1}^{K} \sum_{i=1}^{M_k} (C_{\max} - \sum_{j=1}^{J} T_j \times P_{k,j} \times D_{k,j,i}) \times EI_k \tag{18}$$

$$E_{\max} = E_{process} + E_{idle} \tag{19}$$

Equation (1) indicates that the optimization objective minimizes the energy consumption $E_{\max}$. Equation (2) requires $C_{\max}$ to be greater than or equal to the completion time of the last sublot of all lots in the last stage. Equation (3) requires that only one processing machine can be selected for each lot at the same stage. Equation (4) indicates that the sum of the items contained in all sublots from the same lot must equal the number of items contained in this lot. Equation (5) defines that the number of items contained in each sublot of lots is greater than or equal to 0. Equations (6) and (7) represent the value of $W_{j,e}$. Equation (8) shows that the nonempty sublot in the lots is expected to precede the empty sublot. Equation (9) make sure the start processing time is a non-negative number. Equation (10) shows the calculation method of completion time. In Equation (11), the sublot is required to complete the processing of the previous stage before starting the next stage of processing. Equation (12) expresses that the sublots from the same lot are processed in numbered order at each stage. Equation (13) defines $Y_{k,j,j1,i}$ and $Y_{k,j1,j,i}$. They cannot take the value of 1 at the same time. Equations (14) and (15) are dual constraints, similar to Equation (13), emphasize that the machine can only process one lot at a time. Equation (16) indicates that at the same machine, the later lot can be processed only after the previous lot has been processed; otherwise, the equation does not work. Equations (17) and (18) give the calculation method of total process energy and total idle energy, respectively. Equation (19)

shows that the total energy consumption is equal to the sum of the total process energy consumption and the total idle energy consumption.

4. Improved vCCEA for Solving HFSP_ECS Problem

The proposed vCCEA is developed in this section. First, the design motivation and the algorithm framework are illustrated. After that, the components of vCCEA, involving solution encoding and decoding strategies, initialization strategy, cooperative coevolutionary strategy, VND strategy, and the population restart mechanism, are described in detail. Finally, the whole algorithm procedure is given.

4.1. The Motivations and Framework of vCCEA

The HFSP_ECS is a highly complex combinatorial optimization problem. Recall that when solving the HFSP_ECS, three subproblems must be addressed simultaneously: lot sequence, machine assignment and lot split. These subproblems are not independent but highly coupled. That is, if only one subproblem is optimized, the global optimal solution is almost impossible to be obtained. Therefore, the CCEA is employed, which uses the idea of "divide and conquer" to achieve global optimization by optimizing each subproblem as well as implementing the interaction between subproblems. Among three subproblems, the machine assignment is generally addressed by the proposed heuristic rules [35]. This paper still uses heuristic rules to solve this subproblem, and this rule is incorporated into the decoding strategy. For the other two subproblems, two populations are set up, each of which corresponds to a subproblem. These two populations are lot sequence population and lot split population, which represent lot sequence subproblem and lot split subproblem, respectively. In addition, an archive is also created, where a number of references or complete solutions are stored. It aims to establish collaborative relationships among the individuals from lot sequence and lot split populations in the collaborative coevolutionary process.

The whole algorithm consists of an initialization process, cooperative coevolutionary process of two populations, VND processes, and population restart processes. First, the archive and the two populations are initialized. Then the novel cooperative model was used to control the populations for the cooperative coevolution. In the cooperative coevolutionary process, each individual from the population collaborates with one reference randomly selected from the archive to construct a complete solution. This complete solution is perturbed by the VND process to generate new individuals for updating the population and archive. In this process, different neighborhood structures are designed for individuals from different populations. Moreover, a restart strategy is designed for the individuals who have not been updated for several generations in the population to prevent the algorithm falling into local optima. The vCCEA framework is shown in Figure 1.

4.2. Ending and Decoding

4.2.1. Solution Encoding

Recall that this problem contains three subproblems: lot sequence, machine assignment, and lot split. Based on the problem specific characteristics, a two-layer encoding strategy is adopted in this paper. The first layer uses a permutation $\Pi_J = \{\pi_1, \pi_2 \ldots \pi_j \ldots \pi_J\}$ to represent the scheduling order of the lots. Where π_j indicates the lot index and J represents the total number of the lots. Note that a legitimate permutation requires that each lot only appears once [36]. The lot that appears in advance in the permutation is given higher processing priority, and the scheduling order of lots in subsequent stages is determined by the heuristic rules mentioned in the solution decoding. The second layer uses a matrix $Z_{J\times L}$ with J rows and L columns to represent the lot split, where each row represents the segmentation information of a lot. A complete solution consists of two parts: lot permutation and lot split matrix, which can be expressed as $\langle \Pi_J, Z_{J\times L} \rangle$.

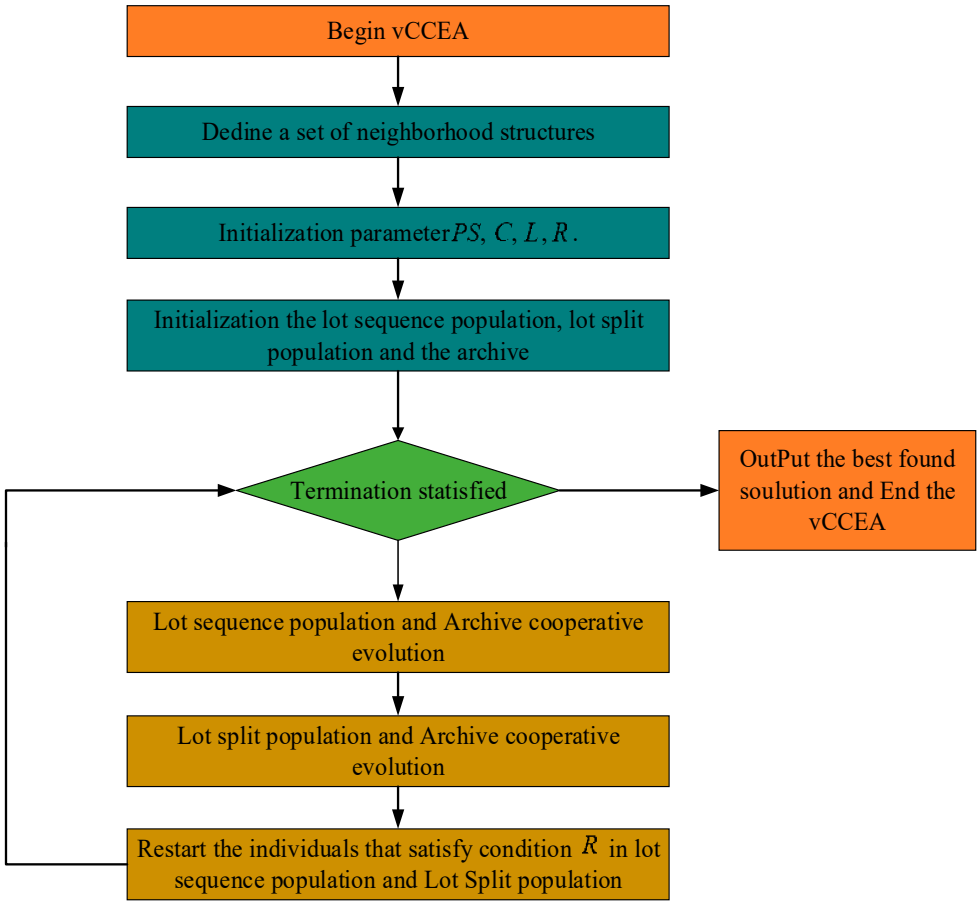

Figure 1. The framework of vCCEA.

Here, a simple illustrative example is given for illustrating the solution encoding. In this example, there are five lots and two stages. The first stage has two parallel machines, i.e., M1 and M2, and the second stage contains three parallel machines, i.e., M3, M4, and M5. The unit idle energy consumption of machines M1 and M2 is 2, and the unit idle energy consumption of machines M3, M4, and M5 also is 2. Lot size, sublot size, item processing time and other specific production data are given in Table 1. According to the above encoding strategy, the encoding for this simple example can be expressed as $\langle \Pi_5, Z_{5\times3} \rangle$, where the first layer is a legal permutation-based encoding vector $\Pi_5 = \{3, 5, 1, 4, 2\}$. This permutation indicates that the current scheduling order is 3,5,1,4,2. In other words, lot 3 was processed first, followed by lot 5, lot 1, lot 4, and lot 2. The matrix $Z_{5\times3}$ serves as the second layer, and is shown in Equation (20). Using lot 5 as an example, the lot 5 is divided into three sublots. The first sublot size is 1, the second sublot size is 1, and the third sublot size is 2.

$$Z_{5\times3} = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 3 & 3 \\ 2 & 2 & 2 \\ 1 & 2 & 2 \\ 1 & 1 & 2 \end{bmatrix} \tag{20}$$

Table 1. Illustrative example of HFSP_ECS.

Lot	Lot Size	Sublot Size			Single Item Process Time		Energy Consumption Per Unit Time	
		Sublot1	Sublot2	Sublot3	Stage1	Stage2	Stage1	Stage2
Lot1	5	1	2	2	1	2	3	2
Lot2	8	2	3	3	1	1	4	3
Lot3	6	2	2	2	2	2	2	4
Lot4	5	1	2	2	2	2	3	3
Lot5	4	1	1	2	1	2	1	2

4.2.2. Solution Decoding

The solution decoding can transform the solution into a feasible schedule, and it mainly solves two problems, lot sequence and machine assignment. For machine assignment, we give priority to idle machines, thus, the "first available machine" rule (FAM) [37] is adopted in this paper. Regarding the lot sequence, the lot scheduling order at stage is determined by the permutation $\Pi_J = \{\pi_1, \pi_2 \ldots \pi_j \ldots \pi_J\}$, while the lot sequence of subsequent stages is determined by the "first-come–first-served" rule. That is, the lot completed earlier at the previous stage is given priority to be scheduled at the following stages. In HFSP_ECS, the sublot of a lot can be immediately transported to the downstream stage for processing when the sublot completes the processing at the current stage. Based on this feature, the "first-come–first-served" rule based on sublot preemption is adopted, i.e., the lot whose first sublot completes the processing at the previous stage first has higher priority at the downstream stage. Under this rule, if the completion time of the first sublots of some lots at the previous stage is equal, the completion time of their second sublots is compared, and so on.

According to the above encoding and decoding strategies, the Gantt chart of the schedule for the illustrative example in Section 4.2.1 is shown in Figure 2. Here, the (a, b) represents a sublot, where a is the lot number, b is the sublot number, and then the minimum energy consumption is 555.

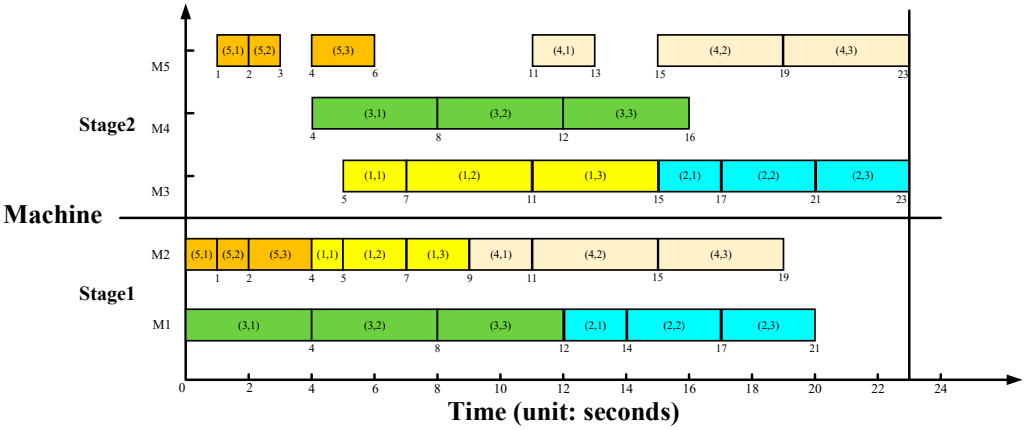

Figure 2. The schedule Gantt chart for the illustrative example.

4.3. Algorithm Initialization

At the beginning of the algorithm, an archive and two collaborative populations need to be initialized. The archive $\left\{\left\langle \Pi^{R[1]}, Z^{R[1]}\right\rangle, \left\langle \Pi^{R[1]}, Z^{R[1]}\right\rangle, \ldots, \left\langle \Pi^{R[PS]}, Z^{R[PS]}\right\rangle\right\}$ is made up of PS combinations. Each combination $\left\langle \Pi^{R[ind]}, Z^{R[ind]}\right\rangle, ind = 1 \ldots PS$ represents a complete solution, where $\Pi^{R[ind]}$ represents the lot sequence for solution ind. Similarly, $Z^{R[ind]}$ represents the lot split for solution ind. When the archive is initialized, the two components of each solution are initialized in two different ways. The lot sequence is

initialized by a random way, while the lot split is determined by the uniform initialization method [38]. The uniform initialization procedure is described as follows.

Procedure Uniform initialization

Step1. Each lot is evenly divided into several sublots. For the jth lot, the size of each sublot is $N_{j,e} = \left\lfloor T_j/L \right\rfloor$,where $\lfloor\ \rfloor$ means the nearest integer that is smaller than T_j/L.
Step2. For the jth, the remaining size r_j is obtained that $r_j = T_j - \sum_{e=1}^{L} N_{j,e}$.
Step3. For the jth, r_j is added to any sublot randomly.

The two populations are the lot sequence population and the lot split population. The lot sequence population consists of PS individuals, i.e., $\left\{\Pi^{[1]}, \Pi^{[2]}, \ldots, \Pi^{[ps]}\right\}$. That is, each individual only represents the lot sequence of a solution, this population is initialized with the lot sequences of the solutions in the archive, i.e., $\Pi^{[ind]} = \Pi^{R[ind]}$ for $ind = 1, 2, \ldots, PS$. Obviously, the individuals in lot sequence population are indeed not the complete solutions, such that they cannot be evaluated directly. To evaluate each $\Pi^{[ind]}$, a collaborator must be identified to build an evaluable solution. Here, the lot split individual $Z^{R[ind]}$ is determined as the collaborator, and the index of this collaborator is recorded by $Col_1[ind] = ind$, where $ind = 1, 2, \ldots, PS$. The individual $\Pi^{[ind]}$ and its collaborator $Z^{[Col_1[ind]]}$ construct a complete solution $\left\langle \Pi^{[ind]}, Z^{[Col_1[ind]]} \right\rangle$. The energy consumption value of the solution is also that of the individual $\Pi^{[ind]}$. Similarly, the lot split population consists of PS lot split matrix initially, i.e., $\left\{Z^{[1]}, Z^{[2]}, \ldots, Z^{[PS]}\right\}$ where $Z^{[ind]}$ for $ind = 1, 2, \ldots, PS$. The individual $\Pi^{R[ind]}$ is determined as the collaborator for $\left\{Z^{[1]}, Z^{[2]}, \ldots, Z^{[PS]}\right\}$, and the index of this collaborator is recorded by $Col_2[ind] = ind$, where $ind = 1, 2, \ldots, PS$. Individual $Z^{[ind]}$ in this population and a lot sequence collaborator $\Pi^{[Col_2[ind]]}$ comprise a new solution $\left\langle \Pi^{[Col_2[ind]]}, Z^{[ind]} \right\rangle$. The energy consumption value of the solution is also that of the individual $Z^{[ind]}$.

4.4. Cooperative Coevolution Process

The whole cooperative coevolutionary process can be divided into two parts: evolution of the lot sequence population and evolution of the lot split population. The evolution of the lot sequence population is first performed. Through this process, individuals in the lot sequence population are updated on the one hand, and certain solutions in the archive can obtain better information of lot sequences on the other hand. Then, the evolution of the lot split population is performed. This process aims to update individuals in the lot split population and ensures that solutions in the archive can obtain better lot split information. Through the above two processes, the evolution of both populations is achieved and the solutions in the archive are also updated during the evolution process. The two processes are described in detail below.

4.4.1. Evolution of the Lot Sequence Population

In the evolution process of the lot sequence population, a complete solution is first constructed by individual $\Pi^{[ind]}$ from lot sequence population and its collaborator Z in the archive. To maintain the diversity of the population and to avoid premature convergence, the last collaborator $Z^{R[Col_1[ind]]}$ of $\Pi^{[ind]}$ that is pointed by the index is not used. Instead, the lot split matrix $Z^{R[rand]}$ is randomly selected from the archive as the current collaborator, where $rand$ is a randomly generated integer between 1 and PS. The combined solution here can be expressed as $\left\langle \Pi^{[ind]}, Z^{R[rand]} \right\rangle$. Then, the VND process is performed on lot sequence $\Pi^{[ind]}$ of the combined solution. A new lot sequence individual $\Pi'^{[ind]}$ is generated by individual $\Pi^{[ind]}$, and the solution $\left\langle \Pi^{[ind]}, Z^{R[rand]} \right\rangle$ comes to $\left\langle \Pi'^{[ind]}, Z^{R[rand]} \right\rangle$.

According to the VND characteristics [39], as long as a new individual $\Pi'^{[ind]}$ is generated, the performance of the new solution $\left\langle \Pi'^{[ind]}, Z^{[rand]} \right\rangle$ needs to be evaluated first. If the objective value of $\left\langle \Pi'^{[ind]}, Z^{[rand]} \right\rangle$ is better than $\left\langle \Pi^{[ind]}, Z^{[rand]} \right\rangle$, then the archive and the population will be updated by a new solution $\left\langle \Pi'^{[ind]}, Z^{[rand]} \right\rangle$. Otherwise, the VND process continues. If the whole VND process fails to find a good $\Pi'^{[ind]}$, then the evolution of the individual is ended for this time.

The process of updating the archive and lot sequence population can be described as follows. If the objective value of the new solution $\left\langle \Pi'^{[ind]}, Z^{R[rand]} \right\rangle$ is better than $\left\langle \Pi^{[ind]}, Z^{R[Col_1[ind]]} \right\rangle$, the solution $\left\langle \Pi^{[ind]}, Z^{R[Col_1[ind]]} \right\rangle$ will be updated by the new solution $\left\langle \Pi'^{[ind]}, Z^{R[rand]} \right\rangle$, i.e., $\Pi^{[ind]} = \Pi'^{[ind]}$, $Col_1[ind] = rand$. Note that the last collaborator $Z^{R[Col_1[ind]]}$ may have been changed in the evolutionary process of the lot split population. At the same time, the archive is attempted to be updated. If the objective value of the new solution $\left\langle \Pi'^{[ind]}, Z^{R[rand]} \right\rangle$ is better than $\left\langle \Pi^{R[rand]}, Z^{R[rand]} \right\rangle$, then the individual $\Pi^{R[rand]}$ will be updated by individual $\Pi'^{[ind]}$, i.e., $\Pi^{R[rand]} = \Pi'^{[ind]}$. The above process is repeated from $ind = 1$ to $ind = PS$. Given the above, the coevolutionary process for the individuals $\Pi^{[ind]}$ in the lot sequence population is shown in Figure 3.

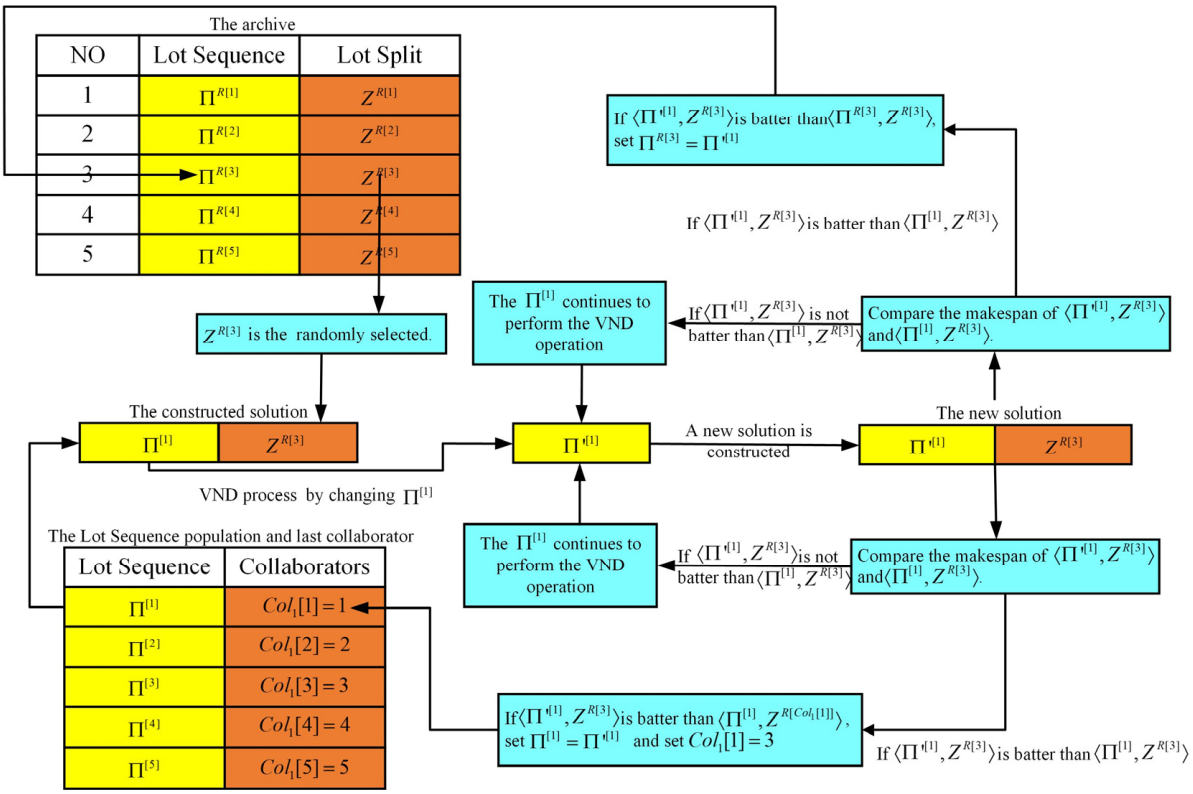

Figure 3. Evolution process of the lot sequence population.

An efficient neighborhood structure plays a key role in the whole algorithm. To design a good neighborhood structure, the problem characteristics must be exploited. When the lot sequence population evolves, the neighborhood structure only works on the lot sequence. That is, a new individual $\Pi'^{[ind]}$ is formed by perturbing the individual $\Pi^{[ind]}$ in the neighborhood structure. Therefore, to obtain a better lot sequence, three neighborhood structures are specially designed based on a solution encoding strategy, and the VND process is used to switch the neighborhoods. Two of these neighborhood structures are

the insert and swap operations that are widely used in HFSP problems, referred to as lot insertion and lot swap. However, when solving the large-scale problems, a single insertion or swap may not effectively perturb the current solution. Therefore, a lot swap operation with a large search range is proposed by improving the lot swap, called the Enhanced lot swap in this instance. The lot insertion is not enhanced here since its high time complexity. The three neighborhood structures for lot sequence are described below: (1) Lot insertion. A new lot sequence is formed by taking a random lot from the lot sequence and inserting it into a randomly different position. (2) Lot swap. Take two lots at random from the lot sequence and exchange their positions in the sequence. (3) Enhanced lot swap. Perform l times of the lot swap, where l is dynamically determined by the number of lots in the lot sequence. We set l as $L \times J$, where L is a real number between 0 and 1, and J is the number of lots. Additionally, the detailed process of lot sequence population evolution is shown in Algorithm 1.

Algorithm 1 Evolution of the lot sequence population

Define a set of neighborhood structures ${N^1}_k, k = 1, \ldots, k_{\max}$
for $ind = 1$ to PS
  $rand \leftarrow$ generate a random integer in $[1 - PS]$
  $\left\langle \Pi^{[ind]}, Z^{R[rand]} \right\rangle \leftarrow$ constitute a complete solution
  Define $\Pi \leftarrow \Pi^{[ind]}$
  Let $k \leftarrow 1$, $Count \leftarrow 0$
  while $k \leq k_{\max}$ do
    while $Count < C$ do
      ${\Pi'}^{[ind]} \leftarrow Neighborhood(\Pi, {N^1}_k)$
      if $\left\langle {\Pi'}^{[ind]}, Z^{R[rand]} \right\rangle$ better than $\left\langle \Pi, Z^{R[rand]} \right\rangle$
        $Count \leftarrow 0$, $k \leftarrow 1$, $\Pi \leftarrow {\Pi'}^{[ind]}$
      if $\left\langle {\Pi'}^{[ind]}, Z^{R[rand]} \right\rangle$ better than $\left\langle \Pi^{[ind]}, Z^{R[Col_1[ind]]} \right\rangle$ or $Z^{R[Col_1[ind]]}$ was changed
        $\Pi^{[ind]} \leftarrow {\Pi'}^{[ind]}, Col_1[ind] = rand$
      end if
      if $\left\langle {\Pi'}^{[ind]}, Z^{R[rand]} \right\rangle$ better than $\left\langle \Pi^{[rand]}, Z^{R[rand]} \right\rangle$
        $\Pi^{[rand]} \leftarrow {\Pi'}^{[ind]}$
      end if
    else
      $Count++$
    end if
  end while
  $k++$
  end while
end for

4.4.2. Evolution of the Lot Split Population

Similar to the evolution process of the lot sequence population, a complete solution is constructed by individual $Z^{[ind]}$ from lot split population and its collaborator $\Pi^{R[rand]}$ in the archive, where *rand* is a randomly generated integer in the range $[1, PS]$. The constructed solution here can be expressed as $\left\langle \Pi^{R[rand]}, Z^{[ind]} \right\rangle$. After that, the VND procedure is executed on the solution $\left\langle \Pi^{R[rand]}, Z^{[ind]} \right\rangle$, and the neighborhood structure here only works on the lot split. Through the VND process, $Z^{[ind]}$ becomes ${Z'}^{[ind]}$, and thus, the solution $\left\langle \Pi^{R[rand]}, Z^{[ind]} \right\rangle$ comes to $\left\langle \Pi^{R[rand]}, {Z'}^{[ind]} \right\rangle$. In the process, as long as a new solution $\left\langle \Pi^{R[rand]}, {z'}^{[ind]} \right\rangle$ is generated, the new solution $\left\langle \Pi^{R[rand]}, {Z'}^{[ind]} \right\rangle$ is evaluated. If $\left\langle \Pi^{R[rand]}, {Z'}^{[ind]} \right\rangle$ is better than $\left\langle \Pi^{R[rand]}, Z^{[ind]} \right\rangle$, then the solution $\left\langle \Pi^{R[rand]}, {Z'}^{[ind]} \right\rangle$

is used to update the archive and lot split population. Otherwise, the VND process continues to find a potentially better solution. The process of updating the archive and lot split population can be described as follows: If the solution $\left\langle \Pi^{R[rand]}, Z'^{[ind]} \right\rangle$ is better than $\left\langle \Pi^{R[Col_2[ind]]}, Z^{[ind]} \right\rangle$, then the $Z^{[ind]}$ and the $Col_2[ind]$ will be updated by $Z'^{[ind]}$ and $\Pi^{R[rand]}$, i.e., $Z^{[ind]} = Z'^{[ind]}$, $Col_2[ind] = rand$. It should also be noted that $\Pi^{R[Col_2[ind]]}$ may have been changed as the lot sequence population evolves. At the same time, if the objective value of the solution $\left\langle \Pi^{R[rand]}, Z'^{[ind]} \right\rangle$ is better than $\left\langle \Pi^{R[rand]}, Z^{R[rand]} \right\rangle$, set $Z^{R[rand]} = Z'^{[ind]}$. This process is shown in Figure 4.

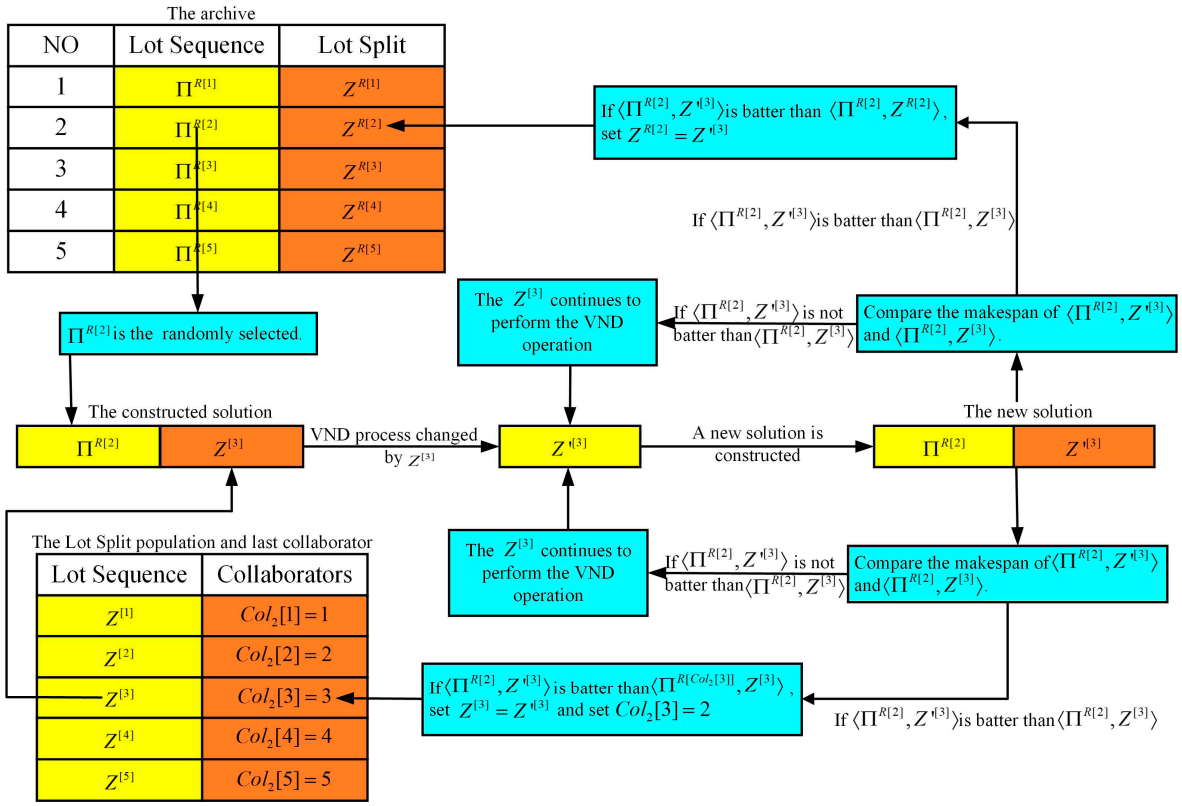

Figure 4. Evolution process of the lot split population.

In the evolution process of the lot split population, to obtain high-quality individuals $Z'^{[ind]}$, three neighborhood structures acting only on the lot split part are specially designed, and the VND process is used to switch the neighborhood. The three perturbation strategies for lot split are described below: (1) Lot split mutation. As shown in Figure 5, from the lot split matrix, a lot with two or more sublots are randomly selected. Reduce a random number in distribution U [1,5] from the size of one sublot and add the number to the size of another sublot. (2) Enhanced lot split mutation. Perform l times lot split mutation, where l is dynamically determined by the number of lots. We set l as $L \times J$, where L is a real number between 0 and 1, and J is the number of lots. (3) Stochastic splits. All lots are redivided into sublots in a random manner. The procedure of the evolution of the lot split population is given in Algorithm 2.

Algorithm 2 Evolution of the lot split population

Define a set of neighborhood structures $N^2{}_k, k = 1, \ldots, k_{\max}$
for $ind = 1$ to PS
  $rand \leftarrow$ generate a random integer in $[1 - PS]$
  $\left\langle \Pi^{R[rand]}, Z^{[ind]} \right\rangle \leftarrow$ constitute a complete solution
  Define $Z \leftarrow Z^{[ind]}$
  Let $k \leftarrow 1$, $Count \leftarrow 0$
  while $k \leq k_{\max}$ do
    while $Count < C$ do
      $Z'^{[ind]} \leftarrow Neighborhood(Z, N^2{}_k)$
      if $\left\langle \Pi^{R[rand]}, Z'^{[ind]} \right\rangle$ better than $\left\langle \Pi^{R[rand]}, Z \right\rangle$
        $Count \leftarrow 0$, $k \leftarrow 1$, $Z \leftarrow Z'^{[ind]}$
      if $\left\langle \Pi^{R[rand]}, Z'^{[ind]} \right\rangle$ better than $\left\langle \Pi^{R[Col_2[ind]]}, Z^{[ind]} \right\rangle$ or $\Pi^{R[Col_2[ind]]}$ was changed
        $Z^{[ind]} \leftarrow Z'^{[ind]}$, $Col_2[ind] = rand$
      end if
      if $\left\langle \Pi^{R[rand]}, Z'^{[ind]} \right\rangle$ better than $\left\langle \Pi^{R[rand]}, Z^{R[rand]} \right\rangle$
        $Z^{[rand]} \leftarrow Z'^{[ind]}$
      end if
    else
      $Count++$
    end if
  end while
  $k++$
  end while
end for

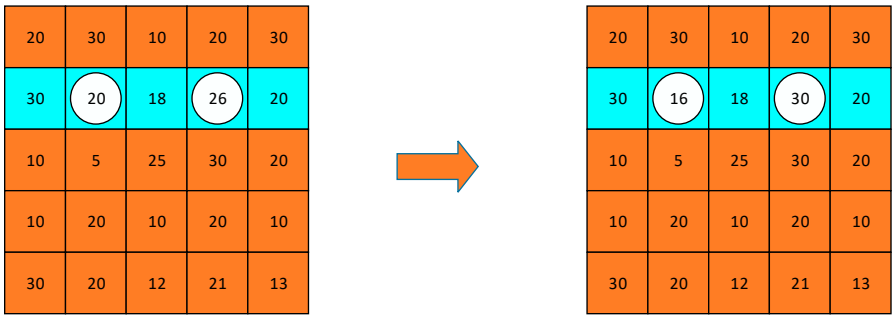

Figure 5. Illustrations of the lot split mutation.

4.5. Coevolutionary Population Restart

With the evolving of the algorithm, the diversity of the two populations might be reduced. In this case, the efficiency of the population coevolution may be poor. To avoid the algorithm falling into the local optimality, two different restart strategies are adopted for the populations. For the lot sequence population, an individual $\Pi^{[ind]}$ is reinitialized if it has not been improved in a predetermined number of R consecutive generations. The novel individual should contain valuable information about the original individual and remain somewhat different from the original individual. For this purpose, a two-point crossover (TPX) method was used, as illustrated in Figure 6. Where two parent lot sequences are randomly selected from the archive because good solutions are stored in an archive. For the lot split population, an individual $Z^{[ind]}$ is also reinitialized if it has not been updated in a predetermined number of R consecutive generations. Due to the lot split matrix is different from the regular sequence, the classical TPX might produce infeasible schedules. Therefore, a cooperative selection operator is proposed. When determining the split information for one lot, two solutions are selected at random from the archive and compared based on their objective values, and the split information for this lot comes from the better one. The

process is repeated from the first lot to the last lot. Here, we use $Z_j^{[ind]}$ to represent the lot split information for lot j in individual ind, and this process is shown in Algorithm 3.

Algorithm 3 Lot split population restart

for $j = 1$ to $j = J$
  Randomly select two solutions in archive $\langle \Pi^a, Z^a \rangle$ and $\left\langle \Pi^b, Z^b \right\rangle$
  if $\langle \Pi^a, Z^a \rangle$ better than $\left\langle \Pi^b, Z^b \right\rangle$
    $Z_j^{[ind]} \leftarrow Z_j^a$
  else
    $Z_j^{[ind]} \leftarrow Z_j^b$
  end if
end for

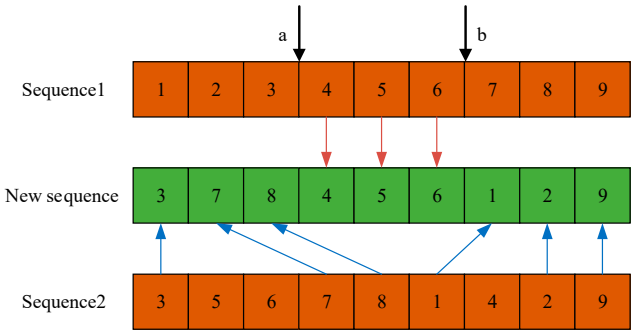

Figure 6. Illustration of TPX for a lot sequence.

4.6. The Algorithm Procedure

With the above description, the whole vCCEA is displayed in Algorithm 4. Where the $UpdateBestSolution(\langle \Pi, Z \rangle)$ means update the optimal solution using solution $\langle \Pi, Z \rangle$, and the $Age(\langle \Pi, Z \rangle)$ represents the number of consecutive update failures of the solution composed of individual Π (or Z) and their collaborators.

Algorithm 4 Lot split population restart

Initialize algorithm parameters, including PS, C, L, R.
Define the termination criterion T.
Define a set of neighborhood structures $N_k^1, k = 1, \ldots, k_{\max}$ and $N_k^2, k = 1, \ldots, k_{\max}$
Initialize archive and two populations
Find the best solutions $\left\langle \Pi^{best}, Z^{best} \right\rangle$ in archive
while T is not satisfied do
  for $ind = 1$ to PS
    $rand \leftarrow$ generate a random integer in $[1 - PS]$
    $\left\langle \Pi^{[ind]}, Z^{R[rand]} \right\rangle \leftarrow$ constitute a complete solution
    Define $\Pi \leftarrow \Pi^{[ind]}$
    Let $k \leftarrow 1$, $Count \leftarrow 0$
    while $k \leq k_{\max}$ do
      while $Count < C$ do
        $\Pi'^{[ind]} \leftarrow Neighborhood(\Pi, N^1{}_k)$
        if $\left\langle \Pi'^{[ind]}, Z^{R[rand]} \right\rangle$ better than $\left\langle \Pi, Z^{R[rand]} \right\rangle$
          $UpdateBestSolution(\left\langle \Pi'^{[ind]}, Z^{R[rand]} \right\rangle)$
          $Count \leftarrow 0$, $k \leftarrow 1$, $\Pi \leftarrow \Pi'^{[ind]}$
          if $\left\langle \Pi'^{[ind]}, Z^{R[rand]} \right\rangle$ better than $\left\langle \Pi^{[ind]}, Z^{R[Col_1[ind]]} \right\rangle$

$\Pi^{[ind]} \leftarrow \Pi'^{[ind]}$, $Col_1[ind] = rand$
$Age\left(\left\langle \Pi^{[ind]}, Z^{R[Col_1[ind]]} \right\rangle\right) \leftarrow 0$
else
$Age\left(\left\langle \Pi^{[ind]}, Z^{R[Col_1[ind]]} \right\rangle\right) ++$
end if
if $\left\langle \Pi'^{[ind]}, Z^{R[rand]} \right\rangle$ better than $\left\langle \Pi^{[rand]}, Z^{R[rand]} \right\rangle$
$\Pi^{[rand]} \leftarrow \Pi'^{[ind]}$
end if
else
$Count++$
end if
end while
$k++$
end while
end for
for $ind = 1$ to PS
$rand \leftarrow$ generate a random integer in $[1 - PS]$
$\left\langle \Pi^{R[rand]}, Z^{[ind]} \right\rangle \leftarrow$ constitute a complete solution
Define $Z \leftarrow Z^{[ind]}$
Let $k \leftarrow 1$, $Count \leftarrow 0$
while $k \leq k_{\max}$ do
while $Count < C$ do
$Z'^{[ind]} \leftarrow Neighborhood(Z, N^2{}_k)$
if $\left\langle \Pi^{R[rand]}, Z'^{[ind]} \right\rangle$ better than $\left\langle \Pi^{R[rand]}, Z \right\rangle$
$UpdateBestSolution\left(\left\langle \Pi^{R[rand]}, Z'^{[ind]} \right\rangle\right)$
$Count \leftarrow 0$, $k \leftarrow 1$, $Z \leftarrow Z'^{[ind]}$
if $\left\langle \Pi^{R[rand]}, Z'^{[ind]} \right\rangle$ better than $\left\langle \Pi^{R[Col_2[ind]]}, Z^{[ind]} \right\rangle$
$Z^{[ind]} \leftarrow Z'^{[ind]}$, $Col_2[ind] = rand$
$Age\left(\left\langle \Pi^{R[Col_2[ind]]}, Z^{[ind]} \right\rangle\right) \leftarrow 0$
else
$Age\left(\left\langle \Pi^{R[Col_2[ind]]}, Z^{[ind]} \right\rangle\right) ++$
end if
if $\left\langle \Pi^{R[rand]}, Z'^{[ind]} \right\rangle$ better than $\left\langle \Pi^{R[rand]}, Z^{R[rand]} \right\rangle$
$Z^{[rand]} \leftarrow Z'^{[ind]}$
end if
else
$Count++$
end if
end while
$k++$
end while
end for
for $ind = 1$ to PS
if $Age\left(\left\langle \Pi^{[ind]}, Z^{R[Col_1[ind]]} \right\rangle\right) > R$
$Restart1(\Pi^{[ind]})$, $Age\left(\left\langle \Pi^{R[Col_2[ind]]}, Z^{[ind]} \right\rangle\right) \leftarrow 0$
end if
if $Age\left(\left\langle \Pi^{R[Col_2[ind]]}, Z^{[ind]} \right\rangle\right) > R$
$Restart2(Z^{[ind]})$, $Age\left(\left\langle \Pi^{R[Col_2[ind]]}, Z^{[ind]} \right\rangle\right) \leftarrow 0$
end if
end for
end while
Output the best solution $\left\langle \Pi^{best}, Z^{best} \right\rangle$.

5. Experimental Analyses

In this section, the performance of the proposed vCCEA is evaluated by experimental design and results analysis. The simulation experiment environment of this paper is a PC with 3.60 GHz Intel Core i7 processor and 32 GB RAM. The vCCEA and all compared algorithms are written in the Visual Studio 2019 C++, and run on the release x64 platform. In the algorithm test, the maximum running time is used as the algorithm termination to ensure fairness. In addition, it is considered that the algorithm has practical significance only when it can solve the problem in an acceptable time. Therefore, the termination condition is set as $t \times J \times K$ milliseconds, where J indicates the number of lots and K represents the number of stages, respectively. Referring to the literature [30], t is set as 80.

5.1. Experimental Dataset and Performance Indicators

In this paper, two benchmark sets β_1 and β_2 are designed to verify the validity of the vCCEA. Where 48 small-scale instances are designed in β_1 to study the difference between the MILP and the vCCEA in solving HFSP_ECS, and 100 medium–large scale instances solved by the metaheuristic algorithm are designed in β_2 to verify the performance of vCCEA. In β_1, the number of lots is J comes from $\{6, 8, 10, 12, 14\}$, and the number of stages S comes from $\{3, 5, 8\}$. Thus, there are 15 different combinations of $J \times S$ that can be obtained. In β_2, the number of lots in J comes from $\{20, 40, 60, 80, 100\}$, and the number of stages is S comes from $\{3, 5, 8, 10\}$. Similarly, there are 20 different combinations in β_2. For β_1, only one instance is randomly generated per combination. For β_2, five instances are randomly generated per combination. Thus, there are 15 small scale instances and 100 medium–large scale instances in β_1 and β_2, respectively. In β_1 and β_2, the number of parallel machines at each stage is randomly generated from the range $[1, 5]$. In addition, the number of items for each lot is obtained from a uniform distribution $U[50, 100]$, the processing time of items at each stage is randomly sampled from the uniform distribution $U[1, 10]$, the processing energy consumption per unit time is obtained from the range $[2, 5]$, the energy consumption per idle unit of the machine takes a value in the range $[1, 3]$, and the maximum number of sublots is set as 5. In this study, time is measured in seconds, and energy consumption is measured in joules, and the relative percentage increase (RPI) is used as the performance metric. The RPI is calculated as in Equation (21).

$$RPI = \frac{E_{avg} - E_{best}}{E_{best}} \times 100 \tag{21}$$

where E_{avg} is the average energy consumption of an instance solved by the given algorithm independently performed several times, and E_{best} is the best result obtained by all the compared algorithms. Algorithms with smaller RPI values will have better performance.

5.2. Parameter Setting

Appropriate parameters are very important to the metaheuristics, which can effectively improve their efficiency and robustness. There are four parameters in the vCCEA proposed in this paper, including the number of solutions in the archive (PS), the maximum number of consecutive failures in a neighborhood during the VND process (C), the parameters (L) that control the number of executions in the two enhanced neighborhood structures and the maximum number of successive generations (R) of updating the individual in two populations unsuccessfully. We first determine the value level of each parameter through preliminary experiments, where the details of value levels are shown in Table 2. To verify the influence of each parameter and its value at different levels, an orthogonal array $L16$ is designed using the Taguchi experimental method to determine their combinations and is displayed in Table 3. For the combinations in Table 3, five instances with different scale problems are selected from benchmark set β_2, and the five different problem scales are 20×5, 40×5, 60×5, 80×5, and 100×5. Each instance is run independently 20 times, and the RPI value of each instance is calculated. Then, as shown in Table 3, the average RPI value of the five instances in each combination is collected as the response value.

Table 2. Parameter level factor.

Parameters	The Level of Parameter			
	1	**2**	**3**	**4**
PS	5	10	15	20
C	5	10	15	20
L	0.1	0.2	0.3	0.4
R	50	100	150	200

Table 3. Orthogonal array and response values.

Combination	Parameter				Response (RPI)
	PS	***C***	***L***	***R***	
1	5	5	0.1	50	0.084
2	5	10	0.2	100	0.0711
3	5	15	0.3	150	0.0384
4	5	20	0.4	200	0.0653
5	10	5	0.2	150	0.0688
6	10	10	0.1	200	0.0766
7	10	15	0.4	50	0.0406
8	10	20	0.3	100	0.0480
9	15	5	0.3	200	0.0806
10	15	10	0.4	150	0.0601
11	15	15	0.1	100	0.0884
12	15	20	0.2	50	0.0623
13	20	5	0.4	100	0.0706
14	20	10	0.3	50	0.0664
15	20	15	0.2	200	0.0748
16	20	20	0.1	150	0.0803

The trend of the parameter level is shown in Figure 7, and Table 4 gives the significance rank of each parameter of the vCCEA. According to Figure 7 and Table 4, it can be concluded that parameter L has the greatest impact on the algorithm among these parameters. This is because the parameter L is related to the VND strategy. In the process of VND, the good or bad neighborhood structure has an important influence on the algorithm. A good neighborhood structure can promote the exploration ability of the algorithm and speed up the convergence. For parameter PS, a larger population size can accommodate more potential solutions and help the algorithm search globally. However, it does not support longitudinal and deep search in a limited running time. Too small a population size is not conducive to global search. For parameter C, a too small value will not make full use of each neighborhood perturbation strategy and a too large value will waste the computational time. For parameter R, if the value is set too small, the good information of advanced individuals in the two populations cannot be fully utilized, and if the value is set too large, the diversity of the population cannot be guaranteed and the algorithm may converge prematurely. Therefore, the appropriate parameters are critical for vCCEA. Through the above parameter experiment and analysis, the best parameter combination we can obtain is $PS = 10$, $C = 15$, $L = 0.3$ and $R = 150$. This parameter combination is used in the following experiments.

5.3. Evaluation of the Algorithm Components and Strategies

In this subsection, the algorithm components and strategies are validated and analyzed for effectiveness. Our algorithm contains the VND process, collaborative model, and two enhanced neighborhood structures, and these three strategies are not independent but highly coupled. To verify the effect of each of the three components and the cooperation between them, three other versions of vCCEA are constructed, vCCEA_1, vCCEA_2, and vCCEA_3, where the vCCEA_1 is the vCCEA that removes the VND process. The vCCEA2

is used to verify the validity of the collaboration model. The VCCEA_3 is the vCCEA that remove the two enhanced neighborhood structures during the VND process. And the β_2 is used to verify vCCEA and the other three versions of vCCEA. For each instance in β_2, the four algorithms are independently run 20 times. For each algorithm, the RPI value of each instance is obtained first. Then, the average RPI of five instances from the same scale problem is calculated and represented by the average RPI values (ARPI). The results are shown in Table 5.

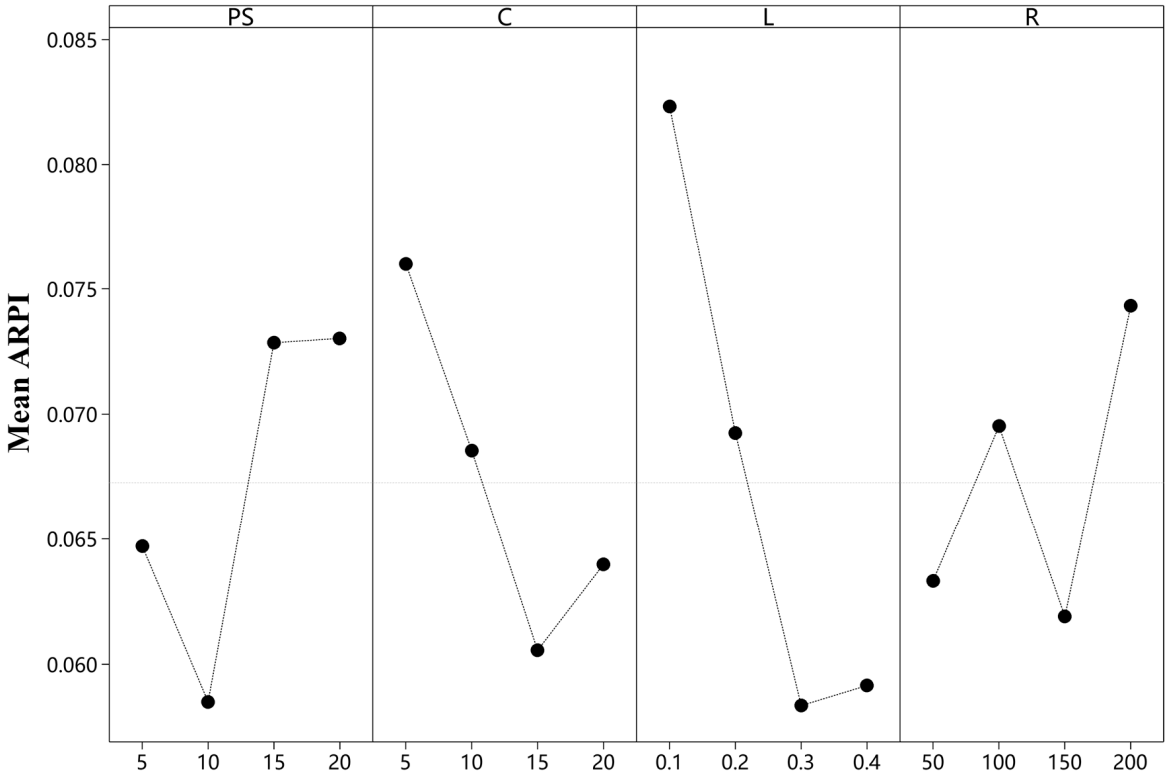

Figure 7. The trend of the parameter level.

Table 4. The average RPI response values.

Level	*PS*	*C*	*L*	*R*
1	0.0647	0.0760	0.0823	0.0633
2	0.0585	0.0686	0.0693	0.0695
3	0.0729	0.0606	0.0584	0.0619
4	0.0730	0.0640	0.0592	0.0743
Delta	0.0145	0.0155	0.0240	0.0124
Rank	3	2	1	4

From Table 5, we can clearly see that the whole vCCEA is the best performer. In the other three versions of the vCCEA, the vCCEA_1 is the worst one. In the cooperative coevolutionary process, the VND process is used to generate new individuals Π(or Z). With the same perturbation strategy, the result of neighborhood switching using VND process is obviously better than that of traditional neighborhood perturbation. Thus, the VND process is crucial to the algorithm. Among these 20 problems with different sizes, the vCCEA_2 is worse than the vCCEA. It can be seen that the collaborative model proposed in this paper is effective. By comparing vCCEA_3 and vCCEA, the validity of the two enhanced neighborhood structures is proven. These two enhanced neighborhood structures can enlarge the search area of the VND process, and it is beneficial for the vCCEA to find the potential promising solution in a larger solution space.

Table 5. Comparison of vCCEA components.

ARPI	vCCEA	vCCEA_1	vCCEA_2	vCCEA_3
20_3	**0.0716**	0.4028	0.5022	0.2135
20_5	**0.0328**	0.111	0.0713	0.1105
20_8	**0.0283**	0.0671	0.0195	0.0553
20_10	**0.0052**	0.0617	0.0226	0.0487
40_3	**0.0251**	0.0841	0.0776	0.0988
40_5	**0.02**	0.0961	0.0766	0.1121
40_8	0.0195	0.0523	**0.0148**	0.056
40_10	**0.0167**	0.0501	0.0886	0.0578
60_3	**0.0049**	0.0379	0.0186	0.0431
60_5	**0.0068**	0.0582	0.0272	0.0597
60_8	**0.0148**	0.0623	0.0668	0.0707
60_10	**0.0083**	0.0571	0.017	0.0506
80_3	**0.0032**	0.0297	0.0085	0.0273
80_5	**0.0142**	0.0556	0.038	0.0549
80_8	**0.0104**	0.044	0.0141	0.0438
80_10	**0.0193**	0.0479	0.0321	0.0465
100_3	**0.0141**	0.0672	0.041	0.0649
100_5	**0.0204**	0.0631	0.0444	0.0532
100_8	**0.0192**	0.0773	0.0318	0.0749
100_10	**0.0189**	0.0416	0.0278	0.0371
Mean	**0.0187**	0.0784	0.062	0.069

5.4. Evaluation of the vCCEA on the Small-Scale Instances

This section focuses on the differences between vCCEA and MILP when solving the small-scale problems. We use the Gurobipy 9.1.2 optimizer to run the MILP on the instances in β_1, and the maximum running time is limited to 3600 s. In addition, the vCCEA is used to solve the instances in β_1, and the termination condition is set to $80 \times J \times K$. The results are shown in Table 6.

Table 6. The validation results of MILP.

Problem	MILP			vCCEA		
	Objective	Time (s)	RPI	Objective	Time (s)	RPI
6_3	**48,421**	4.36	**0**	**48,421**	**1.453**	**0**
6_5	**169,894**	8.13	**0**	**169,894**	**2.406**	**0**
6_8	**364,151**	10.13	**0**	**364,151**	**3.844**	**0**
8_3	**104,164**	20.06	**0**	**104,164**	**1.921**	**0**
8_5	**220,119**	154.4	**0**	**220,119**	**3.203**	**0**
8_8	**369,027**	46.73	**0**	**369,027**	**5.125**	**0**
10_3	**122,068**	17.41	**0**	**122,068**	**2.406**	**0**
10_5	**223,049**	3600	**0**	**223,049**	**4**	**0**
10_8	**612,361**	3600	**0**	**612,361**	**6.406**	**0**
12_3	**222,509**	3600	**0**	**222,509**	**2.906**	**0**
12_5	**311,630**	3600	**0**	**311,630**	**4.813**	**0**
12_8	**612,660**	3600	**0**	**612,660**	**7.688**	**0**
14_3	**237,372**	3600	**0**	**237,372**	**3.375**	**0**
14_5	281,607	3600	3.678	**271,617**	**5.609**	**0**
14_8	684,055	3600	0.1826	**683,506**	**8.984**	**0**

From Table 6, it can be concluded that for small-scale instances, both the MILP and vCCEA can find optimal solutions. The MILP and vCCEA find the same results for the first 13 instances in β_1. In addition, for instances 14_5 and 14_8, the vCCEA revealed better results. As the complexity of the problem increases, the effectiveness of the MILP is gradually inferior to the vCCEA. For large-scale instances, the MILP model has difficulty in providing a good solution in a short time. As we know that time is a non-negligible factor

in actual production, thus the near-optimal solutions are required in an acceptable time. Therefore, with the increasing size of the instances, the advantages of vCCEA become more and more obvious.

5.5. Evaluation of vCCEA on the Medium–Large Scale Instances

Next, the performance of the proposed vCCEA is evaluated on the medium–large scale instances in β_2. Here, we collected five metaheuristic algorithms for comparisons, namely, CVND [30], GA [40], GAR [24], VMBO [41], and DABC [42], which are those presented for the HFSP in the literature most recently and have been proven to have excellent performance. For the HFSP_ECS, three highly coupled subproblems need to be solved, i.e., lot sequence, machine assignment, and lot split. Due to the specificity of the problem, we retain the original characteristics of each comparison algorithm and modify it to adapt to our problem. All algorithms use the same double-layer encoding and decoding strategies as proposed in this paper, and select the corresponding lot split operator from this article. As these comparison algorithms have been partially changed for adapting to our problem, their parameters are also optimized and adjusted on the original basis by using the DOE method to ensure that these algorithms can play with better performance. For each instance in β_2, each algorithm is run independently 20 times, and the average energy consumption and the ARPI of five instances from the same scale problem are calculated. The experimental results are given in Table 7. Additionally, to more visually demonstrate the differences between these six algorithms, the means and 95% least significant difference (LSD) confidence intervals [43] were analyzed. Figure 8 shows the confidence interval comparisons between vCCEA and each algorithm, and Figure 9 shows the confidence interval comparison among all algorithms, where the X-axis represents the various algorithms and the Y-axis is the ARPI value.

Table 7. Comparison results of vCCEA and other algorithms on β_2.

Problem	vCCEA		CVND		GA		GAR		VMBO		DABC	
	AVG	RPI	AVG	RPI	AVG	RPI	AVG	RPI	AVG	RPI	AVG	RPI
20_3	**104,859.2**	**0.0716**	105,168.8	0.367	105,241.6	0.4366	105,622.4	0.8	105,118.9	0.3194	105,380.9	0.5695
20_5	**286,304.1**	**0.0328**	286,558.3	0.1216	286,650.9	0.154	287,162.9	0.3329	286,555.6	0.1207	286,341.3	0.0458
20_8	556,484.2	0.0376	556,780.2	0.0908	556,746.2	0.0847	557,223.5	0.1705	556,616.3	0.0614	**556,421**	**0.0262**
20_10	**674,734.2**	**0.0135**	675,111.7	0.0695	675,225	0.0863	675,640	0.1478	674,985.4	0.0507	674,824.4	0.0269
40_3	**249,561.5**	**0.0251**	249,702.6	0.0817	249,803.9	0.1223	250,206.8	0.2838	249,746.1	0.0991	249,831.4	0.1333
40_5	**435,346.9**	**0.024**	435,831.7	0.1353	435,944.6	0.1613	436,981.2	0.3994	435,847.7	0.139	436,000.1	0.1741
40_8	**1,040,283**	**0.0228**	1,040,946	0.0865	1,041,001	0.0918	1,042,108	0.1982	1,040,757	0.0683	1,040,905	0.0825
40_10	**1,150,216**	**0.0186**	1,151,282	0.1113	1,151,477	0.1282	1,153,416	0.2969	1,151,298	0.1127	1,151,912	0.1661
60_3	**406,475.4**	**0.0049**	406,556.5	0.0248	406,668.4	0.0524	406,839.1	0.0943	406,672.4	0.0533	406,640	0.0454
60_5	**772,749.8**	**0.0068**	772,993.9	0.0384	773,326.6	0.0815	773,701.2	0.1299	773,183	0.0629	773,111.7	0.0536
60_8	**1,272,264**	**0.0205**	1,272,954	0.0747	1,273,432	0.1123	1,277,052	0.3969	1,273,614	0.1266	1,275,208	0.2519
60_10	**1,781,242**	**0.0082**	1,782,288	0.0669	1,782,747	0.0927	1,784,128	0.1703	1,782,522	0.0801	1,782,387	0.0725
80_3	**538,223.9**	**0.0032**	538,356.1	0.0278	538,429.4	0.0414	538,612.1	0.0753	538,420.2	0.0397	538,330.7	0.0231
80_5	**1,105,906**	**0.0143**	1,106,629	0.0796	1,106,944	0.1081	1,107,842	0.1893	1,106,768	0.0922	1,107,026	0.1155
80_8	**1,666,493**	**0.0105**	1,668,307	0.1193	1,667,385	0.064	1,668,484	0.13	1,667,195	0.0526	1,667,606	0.0772
80_10	**2,381,343**	**0.0193**	2,382,087	0.0506	2,382,973	0.0878	2,385,528	0.1951	2,382,764	0.079	2,383,807	0.1228
100_3	**647,384.2**	**0.0141**	647,558.7	0.041	647,756.7	0.0716	648,072.5	0.1204	647,836.8	0.084	647,836.7	0.084
100_5	**1,225,888**	**0.0203**	1,226,310	0.0548	1,226,695	0.0862	1,228,023	0.1946	1,226,821	0.0965	1,227,563	0.1571
100_8	**2,474,499**	**0.0179**	2,477,750	0.1493	2,476,969	0.1178	2,477,487	0.1387	2,475,958	0.0769	2,476,269	0.0895
100_10	**2,907,226**	**0.0188**	2,908,332	0.0569	2,908,726	0.0704	2,910,371	0.127	2,908,690	0.0692	2,910,773	0.1409
Mean	**1,083,874**	**0.0202**	1,084,575	0.0924	1,084,707	0.1126	1,085,725	0.2296	1,084,568	0.0942	1,084,909	0.1229

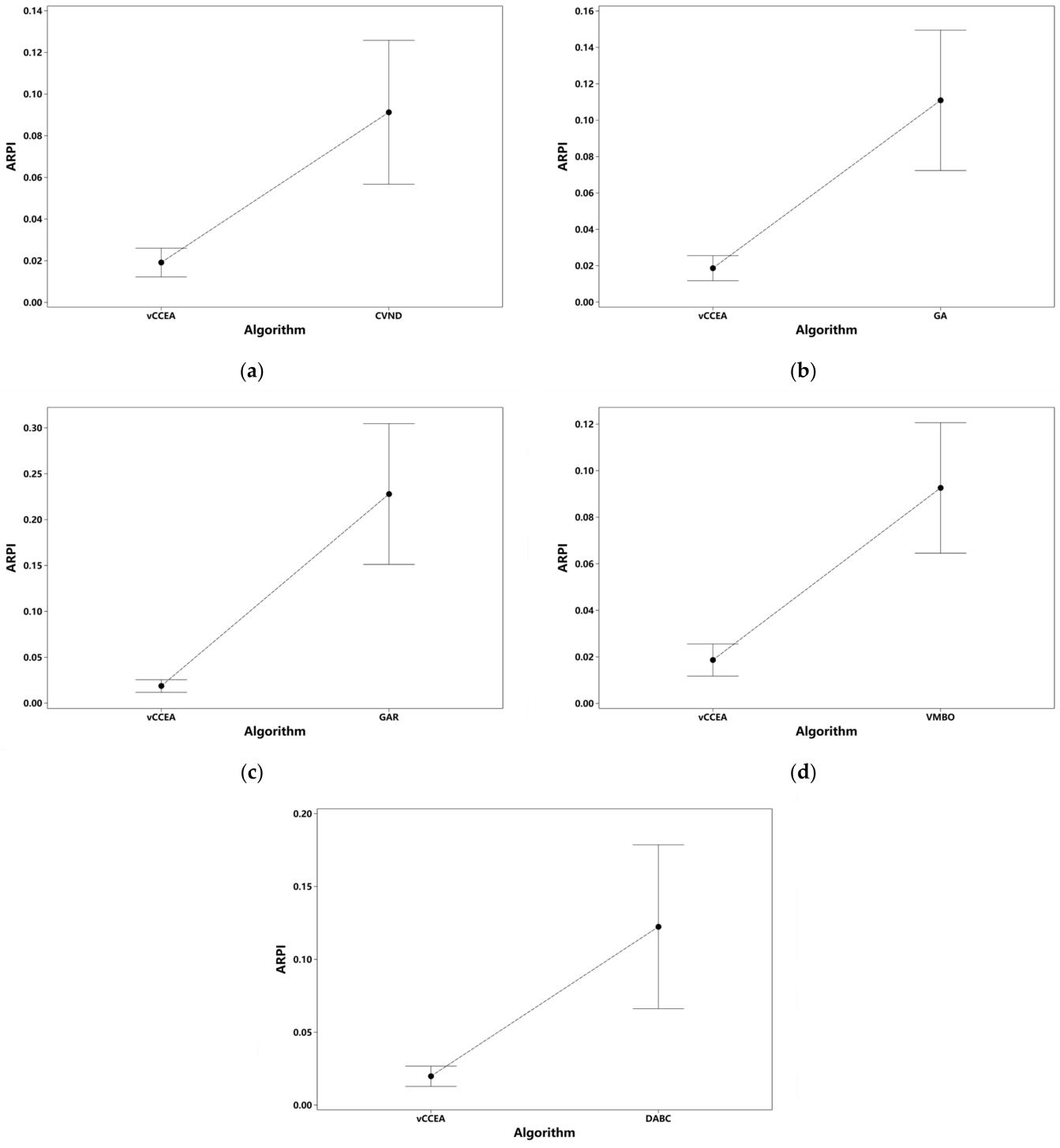

Figure 8. Confidence interval graph. (**a**) Confidence intervals of vCCEA and CVND; (**b**) confidence intervals of vCCEA and GA; (**c**) confidence intervals of vCCEA and GAR; (**d**) confidence intervals of vCCEA and VMBO; (**e**) confidence intervals of vCCEA and DABC.

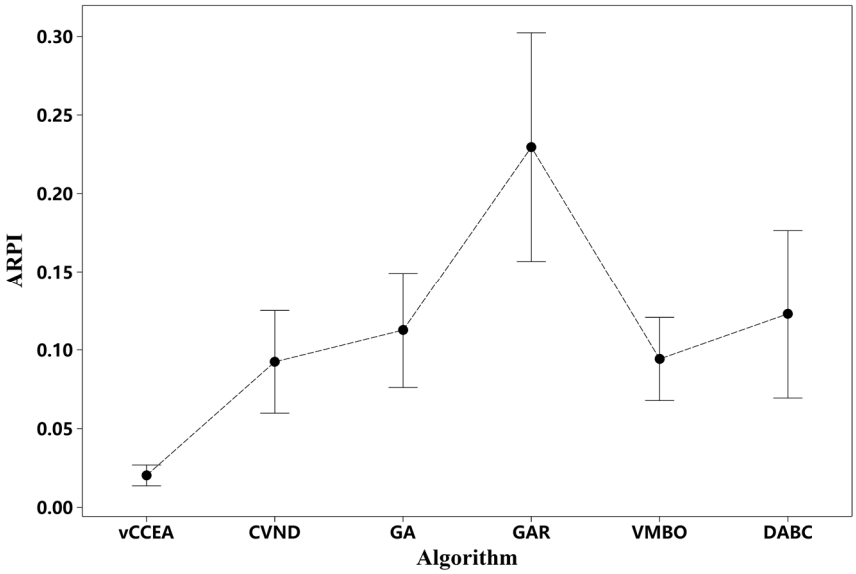

Figure 9. Confidence interval graph for these six algorithms.

As seen in Table 7, the vCCEA is the best one among these algorithms, which obtains best results for 19 of the 20 different problems in β_2. The DABC algorithm finds the optimal solution for the remaining one scale problem, with the scale being 20×8. The last row of Table 7 gives the average energy consumption and average RPI for all instances. It is obvious that vCCEA has the best results among all the algorithms. In addition, according to the confidence intervals shown in Figures 8 and 9, it can clearly be seen that the performance of the proposed vCCEA is obviously better than that of the other five algorithms. To further evaluate the performance of the algorithm, we analyze the convergence of the algorithm, and the convergence curves of these algorithms on two examples are given. These two examples are from 40×5 and 80×10, respectively, and the convergence curves are shown in Figures 10 and 11. The X-axis represents the running time of the algorithm, and the Y-axis represents the energy consumption value.

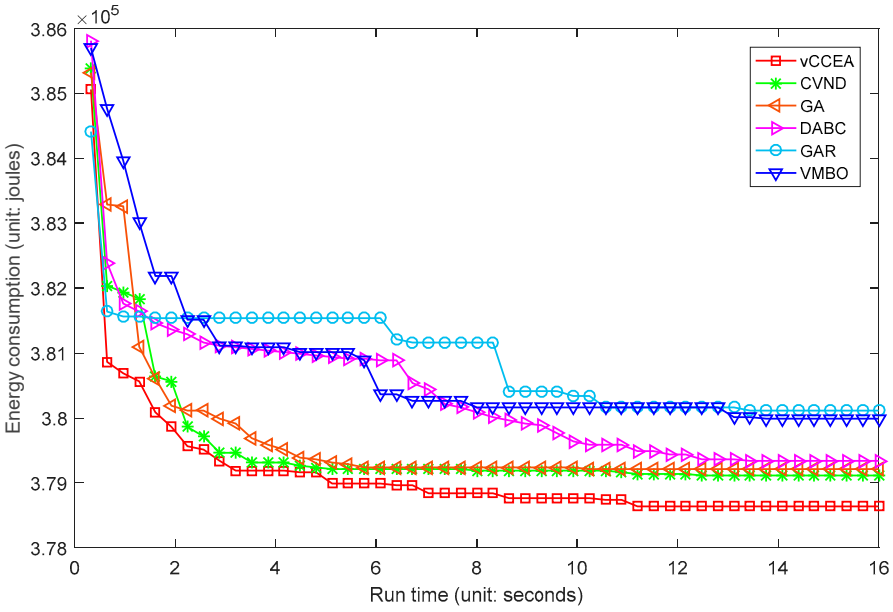

Figure 10. The convergence curve for instances of 40×5.

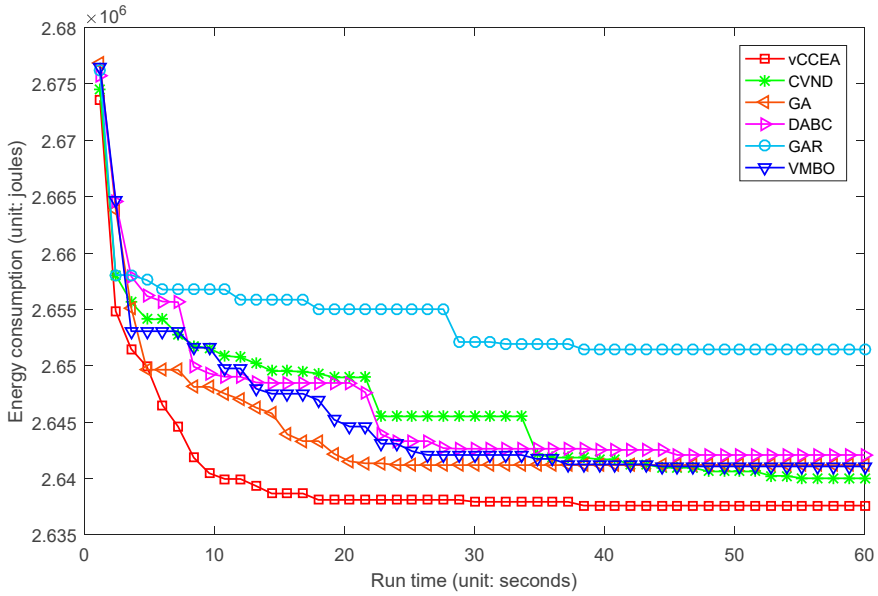

Figure 11. The convergence curve for instances of 80 × 10.

From Figures 10 and 11, the convergence speed of vCCEA is the fastest, and the convergence degree is also better than that of the other algorithms. This is closely related to the cooperative coevolutionary strategy and VND process in the proposed vCCEA. Two populations in the algorithm evolve separately using the VND process specifically designed for them and constantly interacting with the archive. As well as with the population restart strategy, the local search capability of the vCCEA can be ensured, and it also balances the diversity and density of the populations, which further improves the algorithm performance.

Therefore, through the above analysis, the conclusion can be drawn that the vCCEA can effectively solve HFSP_ECS and the robustness of the vCCEA can be guaranteed.

6. Conclusions

In this paper, the energy-efficient flowshop scheduling problem with consistent sublots (HFSP_ECS) is studied, and it supports the overlap of successive operations within a multi-stage manufacturing system. This is a highly complex combinatorial optimization problem that consists of three highly coupled subproblems. We use the minimized energy consumption as the optimization objective. By limiting the maximum number of sublots, a linear integer programming model (MILP) of the addressed problem is established and its validity is verified by the Gurobi optimizer. An improved cooperative coevolutionary algorithm (vCCEA) is proposed by integrating the variable neighborhood decent (VND) strategy. In vCCEA, with the consideration of the problem-specific characteristics, a two-layer encoding strategy is designed, and a novel collaborative interaction model is proposed. Additionally, to ensure the local search ability of the algorithm, different neighborhood structures are designed for different subproblems, and two kinds of enhanced local neighborhood structures are proposed to search for potential promising solutions. To avoid trapping into the local optima, a population restart mechanism is designed. Moreover, through a large number of experiments on different benchmark sets, the effectiveness of the proposed strategies is proved. The experimental results show that vCCEA is significantly better than the mathematical programming and the other algorithms in solving the HFSP_ECS.

For the HFSP_ECS, the maximum sublot quantities is limited in this paper, so in the future, how to divide the lots and the number of sublots is a direction of our research. At the same time, we will consider more production constraints in the future, such as setup, blocking, transportation, and delivery time. In addition, the realistic manufacturing processes always have multi-objective characteristics and variability. This requires us

to consider more optimization objectives and weigh the relationship between multiple objective functions. Furthermore, the possible emergencies during production are also required and studied to derive useful dynamic and rescheduling strategies.

Author Contributions: C.L.: conceptualization, methodology, data curation, software, validation, writing—original draft. B.Z.: conceptualization, methodology, software, validation, writing—original draft. Y.H.: conceptualization, methodology, software, validation, writing—original draft. Y.W.: conceptualization, methodology, supervision, writing—original draft. J.L.: conceptualization, methodology, visualization, investigation. K.G.: conceptualization, methodology, writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China under grant numbers 61803192, 61973203, 62106073, 62173216, and 62173356. We are grateful for Guangyue Youth Scholar Innovation Talent Program support received from Liaocheng University, the Youth Innovation Talent Introduction and Education Program support received from the Shandong Province Colleges and Universities, and the Natural Science Foundation of Shandong Province under grant numbers ZR2021QE195.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author, upon reasonable request.

Conflicts of Interest: The authors declare that they have no conflict of interest.

Abbreviations

Notations

K	Total number of stages.
k	Index of stages, $k \in \{1, 2, \ldots, K\}$.
J	Total number of lots.
j	Index of lots, $j \in \{1, 2, \ldots, J\}$.
M_k	Number of parallel machines at stage k.
i	Index of machines at stage k, $i \in \{1, 2, \ldots, M_k\}$.
T_j	Total number of items of lot j.
L	Maximum number of sublots of each lot.
e	Index of the sublots, $e \in \{1, 2, \ldots, L\}$.
$P_{k,j}$	Item processing time of lot j at stage k.
$EP_{k,j}$	The energy consumption per unit time when lot j is processed on stage k.
EI_k	The energy consumption per unit time when the machine on stage k is idle.
G	A positive large number.

Decision variables

$N_{j,e}$	Number items of sublot e of lot j.
$S_{k,j,e}$	Beginning time of sublot e of lot j at stage k.
$C_{k,j,e}$	Ending time of sublot e of lot j at stage k.
$W_{j,e}$	A binary variable. The value is 1 if items in the sublot e of lot j is greater than 0, and 0 otherwise.
$D_{k,j,i}$	A binary variable. The value is 1 if lot j is scheduled on machine i at stage k, and 0 otherwise.
$Y_{k,j,j1,i}$	A binary variable. When lot j and lot $j1$ are scheduled on the same machine at stage k, the value is 1 if lot j is processed before lot $j1$, and 0 otherwise.
$C_{\max}$	Completion processing time for all lots.
$E_{process}$	Total energy consumption for all machine processing.
E_{idle}	Total energy consumption of all machines when they stay in the idle.
$E_{\max}$	The total energy consumption.

References

1. Ruiz, R.; Vázquez-Rodríguez, J.A. The Hybrid Flow Shop Scheduling Problem. *Eur. J. Oper. Res.* **2010**, *205*, 1–18. [CrossRef]
2. Huang, Y.-Y.; Pan, Q.-K.; Gao, L. An Effective Memetic Algorithm for the Distributed Flowshop Scheduling Problem with an Assemble Machine. *Int. J. Prod. Res.* **2022**. [CrossRef]
3. Peng, K.; Pan, Q.-K.; Gao, L.; Zhang, B.; Pang, X. An Improved Artificial Bee Colony Algorithm for Real-World Hybrid Flowshop Rescheduling in Steelmaking-Refining-Continuous Casting Process. *Comput. Ind. Eng.* **2018**, *122*, 235–250. [CrossRef]
4. Shao, W.; Shao, Z.; Pi, D. Multi-Local Search-Based General Variable Neighborhood Search for Distributed Flow Shop Scheduling in Heterogeneous Multi-Factories. *Appl. Soft Comput.* **2022**, *125*, 109138. [CrossRef]
5. Wu, X.; Cao, Z. An Improved Multi-Objective Evolutionary Algorithm Based on Decomposition for Solving Re-Entrant Hybrid Flow Shop Scheduling Problem with Batch Processing Machines. *Comput. Ind. Eng.* **2022**, *169*, 108236. [CrossRef]
6. Gro, M.; Krumke, S.O.; Rambau, J. *Online Optimization of Large Scale Systems*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2001; pp. 679–704.
7. Neufeld, J.S.; Schulz, S.; Buscher, U. A Systematic Review of Multi-Objective Hybrid Flow Shop Scheduling. *Eur. J. Oper. Res.* **2022**. [CrossRef]
8. Missaoui, A.; Ruiz, R. A Parameter-Less Iterated Greedy Method for the Hybrid Flowshop Scheduling Problem with Setup Times and Due Date Windows. *Eur. J. Oper. Res.* **2022**, *303*, 99–113. [CrossRef]
9. Gong, D.; Han, Y.; Sun, J. A Novel Hybrid Multi-Objective Artificial Bee Colony Algorithm for Blocking Lot-Streaming Flow Shop Scheduling Problems. *Knowl.-Based Syst.* **2018**, *148*, 115–130. [CrossRef]
10. Daneshamooz, F.; Fattahi, P.; Hosseini, S.M.H. Scheduling in a Flexible Job Shop Followed by Some Parallel Assembly Stations Considering Lot Streaming. *Eng. Optim.* **2022**, *54*, 614–633. [CrossRef]
11. Reiter, S. A System for Managing Job-Shop Production. *J. Bus.* **1966**, *39*, 371–393. [CrossRef]
12. Zhang, W.; Liu, J.; Linn, R.J. Model and Heuristics for Lot Streaming of One Job in M-1 Hybrid Flowshops. *Int. J. Oper. Quant. Manag.* **2003**, *9*, 49–64.
13. Cheng, M.; Mukherjee, N.J.; Sarin, S.c. A Review of Lot Streaming. *Int. J. Prod. Res.* **2013**, *51*, 7023–7046. [CrossRef]
14. Wang, W.; Xu, Z.; Gu, X. A Two-Stage Discrete Water Wave Optimization Algorithm for the Flowshop Lot-Streaming Scheduling Problem with Intermingling and Variable Lot Sizes. *Knowl.-Based Syst.* **2022**, *238*, 107874. [CrossRef]
15. Borndörfer, R.; Danecker, F.; Weiser, M. A Discrete-Continuous Algorithm for Globally Optimal Free Flight Trajectory Optimization. In Proceedings of the 22nd Symposium on Algorithmic Approaches for Transportation Modelling, Opti-mization, and Systems 2022, Potsdam, Germany, 8–9 September 2022.
16. Maristany de las Casas, P.; Sedeño-Noda, A.; Borndörfer, R. An Improved Multiobjective Shortest Path Algorithm. *Comput. Oper. Res.* **2021**, *135*, 105424. [CrossRef]
17. Shi, M.; Gao, S. Reference Sharing: A New Collaboration Model for Cooperative Coevolution. *J. Heuristics* **2017**, *23*, 1–30. [CrossRef]
18. Pan, Q.-K.; Gao, L.; Wang, L. An Effective Cooperative Co-Evolutionary Algorithm for Distributed Flowshop Group Scheduling Problems. *IEEE Trans. Cybern.* **2022**, *52*, 5999–6012. [CrossRef]
19. Mladenović, N.; Hansen, P. Variable Neighborhood Search. *Comput. Oper. Res.* **1997**, *24*, 1097–1100. [CrossRef]
20. Liu, J. Single-Job Lot Streaming in M−1 Two-Stage Hybrid Flowshops. *Eur. J. Oper. Res.* **2008**, *187*, 1171–1183. [CrossRef]
21. Cheng, M.; Sarin, S.C.; Singh, S. Two-Stage, Single-Lot, Lot Streaming Problem for a $$1+2$$ 1 + 2 Hybrid Flow Shop. *J. Glob. Optim.* **2016**, *66*, 263–290. [CrossRef]
22. Potts, C.N.; Baker, K.R. Flow Shop Scheduling with Lot Streaming. *Oper. Res. Lett.* **1989**, *8*, 297–303. [CrossRef]
23. Kalir, A.A.; Sarin, S.C. A Near-Optimal Heuristic for the Sequencing Problem in Multiple-Batch Flow-Shops with Small Equal Sublots. *Omega* **2001**, *29*, 577–584. [CrossRef]
24. Naderi, B.; Yazdani, M. A Model and Imperialist Competitive Algorithm for Hybrid Flow Shops with Sublots and Setup Times. *J. Manuf. Syst.* **2014**, *33*, 647–653. [CrossRef]
25. Zhang, P.; Wang, L.; Wang, S. A Discrete Fruit Fly Optimization Algorithm for Flow Shop Scheduling Problem with Intermingling Equal Sublots. In Proceedings of the 33rd Chinese Control Conference, Nanjing, China, 28–30 July 2014; pp. 7466–7471.
26. Zhang, B.; Pan, Q.; Gao, L.; Zhang, X.; Sang, H.; Li, J. An Effective Modified Migrating Birds Optimization for Hybrid Flowshop Scheduling Problem with Lot Streaming. *Appl. Soft Comput.* **2017**, *52*, 14–27. [CrossRef]
27. Zhang, W.; Yin, C.; Liu, J.; Linn, R.J. Multi-Job Lot Streaming to Minimize the Mean Completion Time in m-1 Hybrid Flowshops. *Int. J. Prod. Econ.* **2005**, *96*, 189–200. [CrossRef]
28. Nejati, M.; Mahdavi, I.; Hassanzadeh, R.; Mahdavi-Amiri, N.; Mojarad, M. Multi-Job Lot Streaming to Mini-mize the Weighted Completion Time in a Hybrid Flow Shop Scheduling Problem with Work Shift Con-straint. *Int. J. Adv. Manuf. Technol.* **2014**, *70*, 501–514. [CrossRef]
29. Lalitha, J.L.; Mohan, N.; Pillai, V.M. Lot Streaming in [N-1](1)+N(m) Hybrid Flow Shop. *J. Manuf. Syst.* **2017**, *44*, 12–21. [CrossRef]
30. Zhang, B.; Pan, Q.-K.; Meng, L.-L.; Zhang, X.-L.; Ren, Y.-P.; Li, J.-Q.; Jiang, X.-C. A Collaborative Variable Neighborhood Descent Algorithm for the Hybrid Flowshop Scheduling Problem with Consistent Sublots. *Appl. Soft Comput.* **2021**, *106*, 107305. [CrossRef]
31. Qin, H.-X.; Han, Y.-Y.; Zhang, B.; Meng, L.-L.; Liu, Y.-P.; Pan, Q.-K.; Gong, D.-W. An Improved Iterated Greedy Algorithm for the Energy-Efficient Blocking Hybrid Flow Shop Scheduling Problem. *Swarm Evol. Comput.* **2022**, *69*, 100992. [CrossRef]

32. Duan, J.; Feng, M.; Zhang, Q. Energy-Efficient Collaborative Scheduling of Heterogeneous Multi-Stage Hybrid Flowshop for Large Metallic Component Manufacturing. *J. Clean. Prod.* **2022**, *375*, 134148. [CrossRef]
33. Dong, J.; Ye, C. Green Scheduling of Distributed Two-Stage Reentrant Hybrid Flow Shop Considering Distributed Energy Resources and Energy Storage System. *Comput. Ind. Eng.* **2022**, *169*, 108146. [CrossRef]
34. Geng, K.; Ye, C. A Memetic Algorithm for Energy-Efficient Distributed Re-Entrant Hybrid Flow Shop Scheduling Problem. *IFS* **2021**, *41*, 3951–3971. [CrossRef]
35. Qiao, Y.; Wu, N.; He, Y.; Li, Z.; Chen, T. Adaptive Genetic Algorithm for Two-Stage Hybrid Flow-Shop Scheduling with Sequence-Independent Setup Time and No-Interruption Requirement. *Expert Syst. Appl.* **2022**, *208*, 118068. [CrossRef]
36. Fan, J.; Li, Y.; Xie, J.; Zhang, C.; Shen, W.; Gao, L. A Hybrid Evolutionary Algorithm Using Two Solution Representations for Hybrid Flow-Shop Scheduling Problem. *IEEE Trans. Cybern.* **2021**. [CrossRef] [PubMed]
37. Guinet, A.; Solomon, M.M.; Kedia, P.K.; Dussauchoy, A. A Computational Study of Heuristics for Two-Stage Flexible Flowshops. *Int. J. Prod. Res.* **1996**, *34*, 1399–1415. [CrossRef]
38. Zhang, B.; Pan, Q.; Meng, L.; Lu, C.; Mou, J.; Li, J. An Automatic Multi-Objective Evolutionary Algorithm for the Hybrid Flowshop Scheduling Problem with Consistent Sublots. *Knowl.-Based Syst.* **2022**, *238*, 107819. [CrossRef]
39. Lan, S.; Fan, W.; Yang, S.; Pardalos, P.M. A Variable Neighborhood Search Algorithm for an Integrated Physician Planning and Scheduling Problem. *Comput. Oper. Res.* **2022**, *147*, 105969. [CrossRef]
40. Keskin, K.; Engin, O. A Hybrid Genetic Local and Global Search Algorithm for Solving No-Wait Flow Shop Problem with Bi Criteria. *SN Appl. Sci.* **2021**, *3*, 1–15. [CrossRef]
41. Zhang, X.; Zhang, B.; Meng, L.; Ren, Y.; Meng, R.; Li, J. An Evolutionary Algorithm for a Hybrid Flowshop Scheduling Problem with Consistent Sublots. *Int. J. Autom. Control.* **2022**, *16*, 19–44. [CrossRef]
42. Pan, Q.-K.; Gao, L.; Li, X.-Y.; Gao, K.-Z. Effective Metaheuristics for Scheduling a Hybrid Flowshop with Sequence-Dependent Setup Times. *Appl. Math. Comput.* **2017**, *303*, 89–112. [CrossRef]
43. Balande, U.; Shrimankar, D. A Modified Teaching Learning Metaheuristic Algorithm with Opposite-Based Learning for Permutation Flow-Shop Scheduling Problem. *Evol. Intel.* **2022**, *15*, 57–79. [CrossRef]